Progetto SmartKey

Membri del team di sviluppo:
Massimo Schembri 0000758316
Arianna Solini 0000753359
Riccardo Vasumini 0000753140

Sommario:

Abstract 4

Analisi dei Requisiti 5

Raccolta dei requisiti 5

Analisi del Dominio 6

Vocabolario 6

Sistemi Esterni 7

Analisi Dominio Applicativo 7

Analisi dei Requisiti 8

Requisiti Funzionali 9

Requisiti Non Funzionali 9

Casi d'Uso di SmartKey 9

Scenari di SmartKey 10

Gestione SmartKey 10

Creazione Spazio Dedicato 11

Scelta di Cosa Sincronizzare 12

Sincronizzazione 13

Analisi e Ottimizzazione 14

Uso Spazio Pubblico 15

Uso Spazio Privato 16

Analisi del Rischio 17

Tabella Valutazione dei Beni 17

Tabella Minacce/Controlli 18

Analisi Tecnologica della Sicurezza 19

Security Use Case & Misuse Case 19

Scenari aggiuntivi 21

Gestione Blacklist 21

Security Use Case & Misuse Case Scenari 22

Requisiti di Protezione dei Dati 24

Analisi del problema 25

Analisi Documento dei Requisiti: Analisi delle Funzionalità 25

Tabella Funzionalità 25

Gestione SmartKey: Tabella Informazioni/Flusso 25

Protezione Cartella Utenti: Tabella Informazioni/Flusso 26

Scrittura Log: Tabella informazioni/ Flusso 26

Analisi Documento dei Requisiti: Analisi dei Vincoli 27

Tabella Vincoli 27

Analisi Documento dei Requisiti: Analisi delle Interazioni 27

Tabella Maschere 27

Tabella Sistemi Esterni 28

Analisi Ruoli e Responsabilità 28
Tabella Ruoli 28
Utente: Tabella Ruolo-Informazioni 29
Scomposizione del problema 30
Tabella Scomposizione Funzionalità 30
Gestione Smartkey: Tabella Sotto-funzionalità 30
Gestione Analisi e Ottimizzazione: Tabella Sotto-funzionalità 30
Creazione Modello del Dominio 31
Architettura Logica: Struttura 32
Diagramma dei package 32
Diagramma delle classi: Dominio 32
Diagramma delle classi: InterfacciaBlackList & InterfacciaImpostazioni & InterfacciaHome 33
Diagramma delle classi: GestioneSincronizzazione 33
Diagramma delle classi: InterefacciaAnalisi 34
Diagramma delle classi: AnalisiEOttimizzazione 34
Diagramma delle classi: ProtezioneCartelle 34
Diagramma delle classi: InterfacciaLog & Log 35
Architettura Logica: Interazione 36
Diagramma di sequenza: AggiuntaBlackList 36
Diagramma di sequenza: AggiuntaImpostazione 37
Architettura Logica: Comportamento 38
Diagramma delle attività: Sincronizzazione Cartella 38
Diagramma delle attività: Scan file cartella sorgente 39
Piano di Lavoro 40
Piano del Collaudo 41

Progettazione 43
Progettazione Architetturale 43
Requisiti Non Funzionali 43
Scelta dell'Architettura 43
Progettazione di Dettaglio 46
Struttura 46
Diagramma di Dettaglio : Dominio-FileSystem 46
Diagramma di Dettaglio : Dominio-Blacklist-Utente-IDispositivo-ImpostazioneTrasferimento 47
Diagramma di Dettaglio : Dominio-Log 48
Diagramma di Dettaglio : Interfacce per i controller 49
Diagramma di Dettaglio : Controller 52
Diagramma di Dettaglio : InitControllers e DataPersistence 54
Diagramma di Dettaglio : View 55
Diagramma di dettaglio: view Home SmartKey 56
Diagramma di dettaglio: view Home Impostazioni Trasferimento 57

Diagramma di dettaglio: view HomeBlackList 57
Diagramma di dettaglio: view Analisi e Ottimizzazione 58
Diagramma di dettaglio: view HomeLog 58
Interazione 59
Diagramma di Sequenza : Aggiunta Impostazione 59
Diagramma di Sequenza : Aggiunta Utente in BlackList 60
Comportamento 60
Progettazione della Persistenza 61
XMLSchema: ImpostazioneTrasferimento 61
XMLSchema: Blacklist 62
Formato File Log 63
Progettazione del Collaudo 63
Progettazione del Deployment 64
Implementazione 65
Problematiche 65
Variazioni 65
Blacklist 65
Log 65
Impostazione Trasferimento 65
Sincronizzazione 66
Gestione delle view 66
Collaudo 67
Deployment 68
Artefatti 68
Deployment type-level 68

Abstract

Dal loro arrivo nel mercato i dispositivi di archiviazione USB hanno avuto sempre le stesse funzioni nonostante il cambiamento di esigenze e tecnologie nel mondo dell'informatica.
Al giorno d'oggi, infatti, i dati che risiedono in tali dispositivi non hanno particolari garanzie di protezione di default né hanno un sistema di sincronizzazione intelligente che vada al di là di un copia/incolla.
Il nostro obiettivo è di renderli più sicuri e comodi nell'uso.
Per rendere questo possibile, abbiamo realizzato un software che permette di configurare sincronizzazione e garantire integrità per i file di uno o più utenti che utilizzano lo stesso dispositivo USB.

Analisi dei Requisiti

Raccolta dei requisiti

1. Si deve realizzare un'interfaccia utente più intuitiva e semplice per la gestione di dispositivi di archiviazione USB
2. La funzionalità principale che deve offrire questa interfaccia sarà la sincronizzazione di cartelle tra dispositivo di archiviazione USB e PC
3. Deve essere possibile sincronizzare un insieme di file o cartelle dall'hard disk di un calcolatore al dispositivo di archiviazione usb e viceversa.
4. L'utente sceglie le cartelle che vuole sincronizzare.
5. Il dispositivo rimovibile può contenere dati relativi a più utenti.
6. Deve essere presente una cartella dedicata all'uso senza software.
7. Al di fuori dell'account proprietario la cartella privata deve essere inaccessibile.
8. Le interfacce del programma devono essere intuitive per le funzionalità offerte.
9. L'applicazione deve offrire un servizio di compressione dati.
10. L'applicazione deve essere implementata con il linguaggio C#

Aggiornamento dopo una discussione con il committente:

Riguardo al punto 7 dei requisiti, dopo aver spiegato i motivi per i quali la chiavetta sarebbe inutilizzabile con una cartella privata inaccessibile, il committente ha deciso di rilassare questo requisito.

Ora le cartelle private dovranno essere accessibili in sola lettura da tutti gli utenti ma il contenuto potrà essere modificato solamente dall'utente.

Analisi del Dominio

Vocabolario

Termine	Definizione	Sinonimi
PC(Personal Computer)	Macchina elettronica capace di accettare e elaborare informazioni, per poi fornire i risultati dell'elaborazione sotto forma di dati alfanumerici.	Computer, Calcolatore
Dispositivo di Archiviazione USB	Memoria di massa portatile di dimensioni molto contenute che si collega al computer mediante la porta USB.	Chiave USB, Penna USB
Porta USB	Interfaccia standard di comunicazione.	Presa USB
Cartella	Contenitore di file di qualunque tipo, anche cartelle a sua volta.	
Cartella Sorgente	Cartella della quale si vuole far la sincronizzazione, contiene l'informazione che deve essere replicata.	
Cartella Destinazione	Cartella che dovrà contenere gli stessi file della cartella sorgente una volta avvenuta la sincronizzazione.	
Cartella Pubblica	Cartella disponibile a tutti gli utenti	Cartella Condivisa
Cartella Privata	Cartella privata dell'utente in cui solo lui ha diritti di modifica.	Cartella Dedicata, Cartella Utente
Cartella Utente Pubblica	Cartella, omonima della cartella privata dell'utente, presente all'interno della cartella pubblica per migliorare l'usabilità del software.	Cartella Omonima

Utente	Utente che esegue il software SmartKey.	Account Proprietario
Interfaccia Utente	Grafica del software smartkey.	Interfaccia
Funzionalità	Servizi offerti dal software smartkey.	
Avvio	Avvio del software SmartKey.	
Sincronizzazione	Operazione che consiste nel rendere una cartella o un file uguale a un altro.	Replicazione, Copia
File System	Gerarchia di cartelle che organizza tutti i dati di un calcolatore.	
Compressione Dati	Metodo che consente di ridurre la dimensione dei file.	Compressione
SmartKey	Il nome del software.	Software SmartKey, Applicativo, Sistema

Sistemi Esterni

Il software SmartKey interagisce con due sistemi esterni: Esplora Risorse di Windows e le ACL.
Esplora Risorse di Windows è la shell di esplorazione dei file di Windows. Questa svolge semplicemente la funzione di fornire un'interfaccia grafica che mostri all'utente la struttura del filesystem in modo intuitivo e permetta, altrettanto intuitivamente, di navigarla.
Le Access Control List sono un sistema di controllo per gli accessi e le modifiche possibili su file e directory.

Analisi Dominio Applicativo

Il progetto è basato su 2 funzionalità molto importanti e richieste al giorno d'oggi: la sincronizzazione, la condivisione dei dati tra utenti.
Citiamo per similitudine un'applicazione che è intrecciata a queste 2 e oramai molto diffusa: Dropbox Desktop App. Si tratta di un servizio di sincronizzazione che rende disponibili dati in rete e che è costituito sia da spazi dedicati alla condivisione di questi tra utenti sia uno spazio personale che solo il proprietario dell'account può vedere.
Nel nostro caso, non avendo a disposizione un'infrastruttura remota, ci appoggiamo a un servizio di archiviazione locale(dispositivo USB) che non ci consente di avere una separazione totale degli utenti perché i dati sono sempre presenti sul dispositivo, mentre

nell’altro caso vengono scaricati dall’infrastruttura remota solamente i file di proprietà di quell’utente.

Analisi dei Requisiti

Punto 1-2:
L’obiettivo del progetto è quello di realizzare un’interfaccia utente semplice e intuitiva per la gestione di periferiche USB, con funzionalità principale la sincronizzazione del dispositivo con il filesystem del PC.

Punto 3-4:
Il dispositivo potrà essere utilizzato da più utenti, quindi il software SmartKey si occuperà della creazione delle cartelle private di cui ognuno è proprietario, all’interno delle quali saranno presenti le relative cartelle scelte dall’utente per la sincronizzazione.
Oltre a queste, sulle quali l’account proprietario ha esclusivo diritto di modifica e scrittura, è possibile scegliere altre cartelle che risiedono nell’area pubblica in modo da poter sincronizzare file modificati da altri utenti da computer diversi.

Punto 6:
All’interno della chiavetta deve essere presente uno spazio in cui qualunque utente possa effettuare operazioni comuni(lettura/scrittura/creazione/cancellazione).
In modo da poter effettuare qualunque modifica al file su cui si sta lavorando senza problemi di sicurezza.

Punto 7:
Un utente non può modificare o creare file all’interno di cartelle private non proprie, ma solamente leggerli.
Poiché gli utenti possono lavorare tutti in un’area pubblica condivisa, l’utente verrà avvertito se sta sincronizzando file prodotti da altri utenti, e in quel caso potrà aggiungere quell’utente a una blacklist o accettare i suoi contenuti.
In caso venga messo in una blacklist, tutti i file prodotti dall’utente malevolo, all’interno della cartella utente che non ha accettato i suoi contenuti, saranno cancellati.

Punto 9:
Per garantire un uso ottimale dello spazio di archiviazione il software SmartKey dovrà essere in grado di comprimere cartelle, contenute nella cartella privata, su richiesta dell’utente.

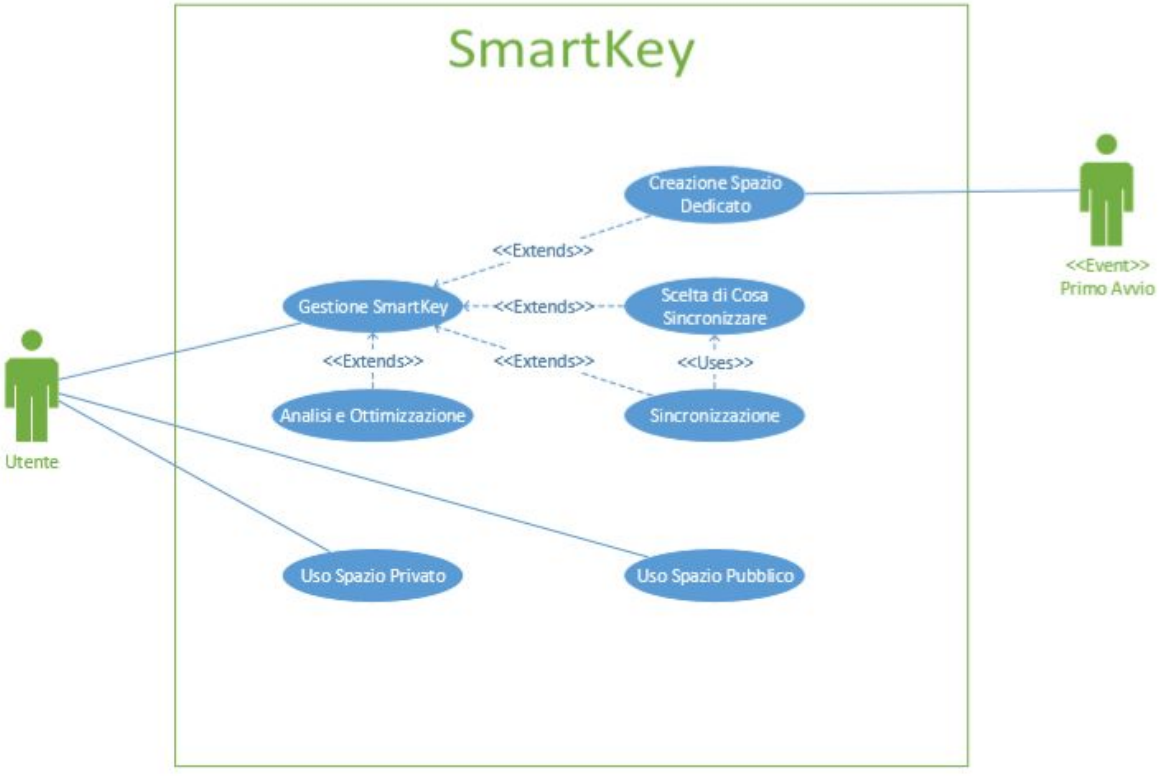
Requisiti Funzionali

- Deve essere possibile sincronizzare il contenuto di una cartella specificata appartenente al PC o al dispositivo di archiviazione USB con la destinazione richiesta dall'utente.Questa operazione di replicazione sarà effettuata ogni qualvolta sarà premuto il tasto dedicato.
- Gestire l'uso del dispositivo di archiviazione USB da parte di più utenti, separando i dati di ogni utente.
- Avere uno spazio condiviso all'interno del dispositivo di archiviazione in cui ogni utente ha accesso e permessi di lettura/scrittura.
- L'utente verrà notificato in caso di sincronizzazione di file non prodotti da lui
- I file di un utente, nella propria cartella privata, devono essere modificabili solo da quell'utente.

Requisiti Non Funzionali

- La sincronizzazione deve essere il più possibile trasparente all'utente
- Le interfacce del software SmartKey devono essere intuitive per le funzionalità offerte
- Il software SmartKey deve essere implementato con il linguaggio C#

Casi d'Uso di SmartKey



Scenari di SmartKey

Gestione SmartKey

Titolo	Gestione SmartKey
Descrizione	Gestione del dispositivo di archiviazione USB e delle sue funzionalità
Attori	Utente, Primo Avvio
Relazioni	Analisi e Ottimizzazione, Scelta di Cosa Sincronizzare, Sincronizzazione, Creazione Spazio Dedicato
Precondizioni	● Il software SmartKey deve essere correttamente installato nel sistema e in funzione ● Il dispositivo di archiviazione USB deve essere correttamente inserito nel PC
Postcondizioni	Sono state eseguite correttamente tutte le funzionalità previste
Scenario principale	● L'utente naviga fino alla schermata di suo interesse. ○ Home SmartKey ○ Impostazioni Trasferimento ○ Gestione BlackList ○ Analisi e Ottimizzazione ○ Log ● Effettua le operazioni disponibili nella schermata
Scenari alternativi	Nel caso il software SmartKey venga avviato per la prima volta verranno eseguite le operazioni previste da Creazione Spazio Dedicato, il quale è scatenato dall'evento Primo Avvio
Requisiti non funzionali	
Punti aperti	

Creazione Spazio Dedicato

Titolo	Creazione Spazio Dedicato
Descrizione	Si occupa di predisporre lo spazio all'interno del dispositivo di archiviazione USB e ad un uso corretto del software SmartKey
Attori	Utente, Primo Avvio
Relazioni	Gestione SmartKey
Precondizioni	• Il software SmartKey deve essere correttamente installato nel sistema e in funzione • Il dispositivo di archiviazione USB deve essere correttamente inserito nel PC
Postcondizioni	Devono essere presenti nel dispositivo di archiviazione USB le tre cartelle specificate nello scenario principale
Scenario principale	1. Si verifica l'evento Primo Avvio. 2. Viene creata una cartella pubblica accessibile a chiunque 3. Viene creata una cartella privata 4. Viene creata una cartella omonima di quella privata nella cartella pubblica.
Scenari alternativi	
Requisiti non funzionali	
Punti aperti	

Scelta di Cosa Sincronizzare

Titolo	Scelta Di Cosa Sincronizzare
Descrizione	L'utente seleziona le cartelle sorgente e destinazione per la sincronizzazione
Attori	Utente
Relazioni	Sincronizzazione, Gestione SmartKey
Precondizioni	• Il software SmartKey deve essere correttamente installato nel sistema e in funzione • Il dispositivo di archiviazione USB deve essere correttamente inserito nel PC • Le cartelle interessate si trovano o sul dispositivo USB o sul PC dell'utente che ha il programma installato.
Postcondizioni	La configurazione delle scelte è stata modificata correttamente
Scenario principale	1. L'utente accede al software SmartKey e raggiunge l'interfaccia di impostazione 2. L'utente si trova davanti una sottofinestra con l'elenco delle cartelle scelte per la sincronizzazione 3. L'utente, tramite Esplora Risorse Windows, naviga il file system per cercare le cartelle sorgente e destinazione che vuole aggiungere alle impostazioni 4. Viene mostrata una nuova configurazione delle scelte 5. L'utente salva le impostazioni
Scenari Alternativi	Scenario a: 1. L'utente vuole rimuovere una o più cartelle da quelle già scelte 2. Si rimanda al punto 1 e 2 dello scenario principale 3. Seleziona l'impostazione che vuole rimuovere nella sottofinestra e, guidato dall'interfaccia, la rimuove 4. Viene ripetuto il punto 3 per ogni rimozione 5. Viene mostrata una nuova configurazione delle scelte 6.L'utente salva le impostazioni
Requisiti non funzionali	
Punti aperti	

Sincronizzazione

Titolo	Sincronizzazione
Descrizione	Trasferimento di dati secondo le impostazioni
Relazioni	Gestione SmartKey, Scelta Di Cosa Sincronizzare
Attori	Utente
Precondizioni	• Il software SmartKey deve essere correttamente installato nel sistema e in funzione • Il dispositivo di archiviazione USB deve essere correttamente inserito nel PC • Sono state impostate le cartelle da sincronizzare
Postcondizioni	Le cartelle selezionate dall'utente e i relativi contenuti saranno replicati sul PC o sul dispositivo di archiviazione USB a seconda dell'impostazione scelta per quella cartella.
Scenario principale	1. L'utente apre il software SmartKey 2. Preme il pulsante dedicato alla sincronizzazione 3. Il sistema controlla che ci siano delle impostazioni valide per l'utente 4. Avvio della sincronizzazione secondo le impostazioni
Scenario alternativo	2. Il sistema dopo aver controllato non trova delle impostazioni valide per quell'utente e si presenterà la schermata di primo avvio.
Requisiti non funzionali	• La sincronizzazione deve essere il più possibile trasparente all'utente
Punti aperti	

Analisi e Ottimizzazione

Titolo	Analisi e Ottimizzazione
Descrizione	Analisi della memoria occupata del dispositivo USB da ciascun utente e compressione a scelta delle cartelle dell'utente (Ottimizzazione)
Attori	Utente
Relazioni	Gestione SmartKey
Precondizioni	• Il software SmartKey deve essere correttamente installato nel sistema e in funzione • Il dispositivo di archiviazione USB deve essere correttamente inserito nel PC
Postcondizioni	
Scenario principale	1. L'utente apre il software SmartKey, raggiunge l'interfaccia di analisi e ottimizzazione e, nel frattempo, il software SmartKey analizza la cartella privata. 2. L'utente si trova davanti a una sottofinestra con l'elenco delle cartelle per le quali è consigliata l'ottimizzazione 3. L'utente, visionato l'elenco dei consigli, può decidere: a. di selezionare e poi comprimere, con l'apposito bottone, alcune cartelle consigliate b. di svuotare la sottofinestra da tutti i consigli riportati dal software SmartKey c. di non svolgere alcuna azione
Scenari alternativi	
Requisiti non funzionali	
Punti aperti	

Uso Spazio Pubblico

Titolo	Uso Spazio Pubblico
Descrizione	Trasferimenti effettuati da un qualsiasi utente nello spazio accessibile da tutti all'interno del dispositivo di archiviazione USB.
Attori	Utente
Relazioni	
Precondizioni	Il dispositivo di archiviazione USB deve essere inserito correttamente al PC.
Postcondizioni	
Scenario principale	1. Un qualsiasi utente accede alla cartella pubblica tramite l'esplora risorse di windows 2. Effettua una qualsiasi operazione(creazione,modifica,lettura) di un file tramite l'esplora risorse di windows. 3. Estrazione della chiavetta
Scenari alternativi	1. La cartella pubblica non è presente perciò viene creata.
Requisiti non funzionali	
Punti aperti	

Uso Spazio Privato

Titolo	Uso Spazio Privato
Descrizione	Operazioni di modifica e/o lettura effettuate nella cartella privata
Attori	Utente
Relazioni	
Precondizioni	Il dispositivo di archiviazione USB deve essere inserito correttamente al PC.
Postcondizioni	
Scenario principale	1. L'utente accede alla propria cartella privata tramite l'esplora risorse di Windows. 2. Effettua una qualsiasi operazione(creazione,modifica,lettura) di un file tramite l'esplora risorse di Windows. 3. Estrazione della chiavetta
Scenari alternativi	Scenario a: 1. La cartella privata non è presente perciò viene creata. Scenario b: 1. L'utente accede ad una cartella privata (non propria) tramite l'esplora risorse di Windows 2. Accede in lettura ai file interessati 3. Estrazione della chiavetta
Requisiti non funzionali	
Punti aperti	

Analisi del Rischio

Tabella Valutazione dei Beni

Bene	Valore	Esposizione
Impostazioni di sincronizzazione	Medio.Contiene le impostazioni relative alla cartella da sincronizzare	Nessuna.
Dati protetti di un utente	Medio.Sono i dati di un utente disponibili in lettura a chiunque ma dei quali bisogna mantenerne l'integrità.	Nessuna
Informazioni FileSystem	Alto.Contiene informazioni relative al filesystem dell'utente	Alto.Perdita di immagine se vengono ricavate informazioni riguardo al filesystem

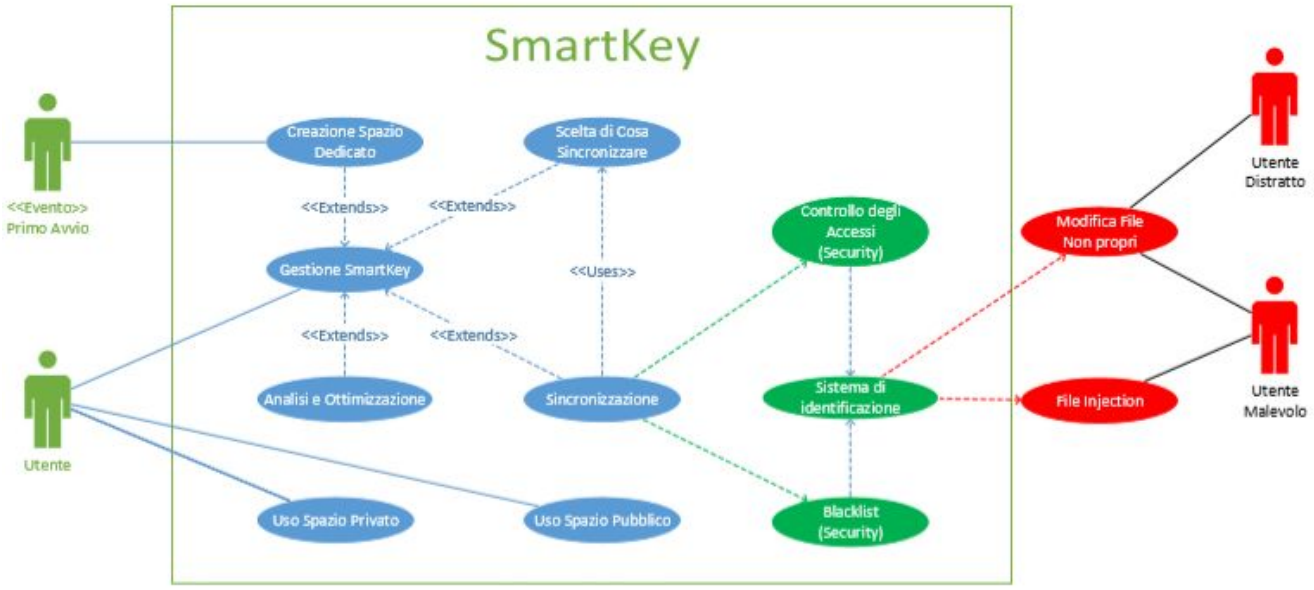
Tabella Minacce/Controlli

Minaccia	Probabilità	Controllo	Fattibilità
Sincronizzazione di file non propri	Alta, poiché la cartella pubblica è scrivibile da tutti gli utenti.	Sistema di blacklist, Log	Basso costo di realizzazione,la decisione di chi inserire nella lista è a carico dell'utente. Quindi in quella fase deve prestare attenzione.
Scrittura di file non propri	Alta, i file sono tutti accessibili.	Sistema di identificazione Access Control List (ACL)	Il sistema operativo le fornisce in modo nativo,consentendo di dare permessi diversi(anche molto specifici) ad ogni utente.
Cambiamento dei permessi relativi a un file/cartella	Alta, tutti i sistemi operativi forniscono un'interfaccia per il cambiamento dei permessi.	Sistema di identificazione Access Control List (ACL)	Il sistema operativo, tramite le ACL, consente di dare/negare il permesso di "cambiamento dei permessi". Nel nostro caso solo l'utente avrà questo permesso.

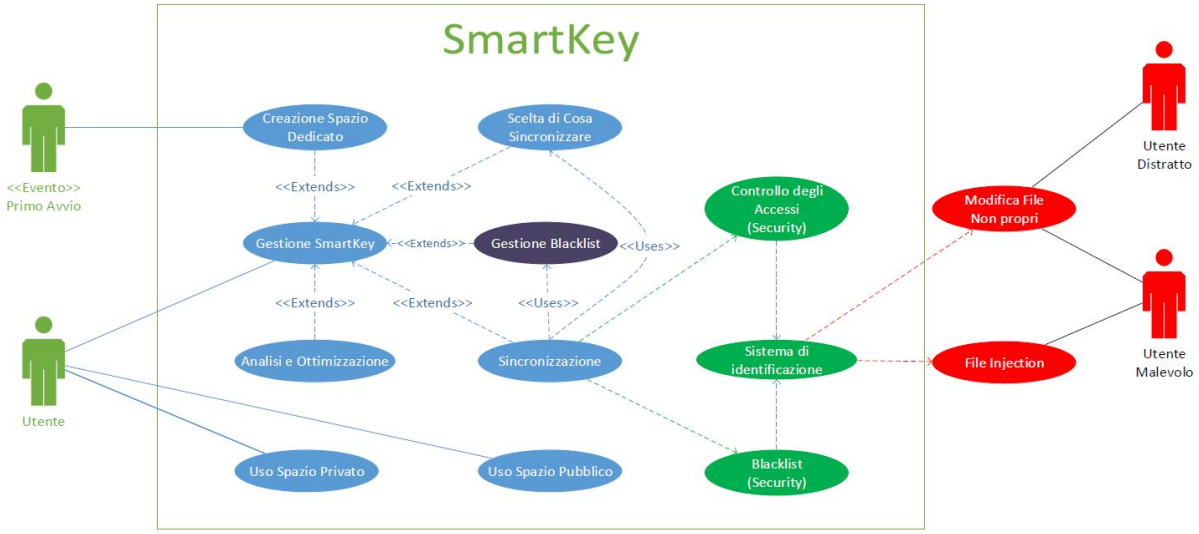
Analisi Tecnologica della Sicurezza

Tecnologia	Vulnerabilità
Sistema di identificazione	• È possibile aggirarlo conoscendo le credenziali di accesso.
Access Control List (ACL)	• Possono essere modificabili, se non correttamente impostate • Uso di permessi poco restrittivi • Se l'utente lascia la propria postazione incustodita, un attaccante potrebbe cambiare i permessi relativi alla cartella privata
Blacklist	• Un attaccante potrebbe creare utenti con nomi simili a quelli che l'utente potrebbe accettare • Creazione di utenti multipli che inseriscono un file.Al momento della sincronizzazione all'utente saranno mostrati molti pop-up e quest'ultimo accetterà tutti gli utenti.

Security Use Case & Misuse Case



Il sistema di sicurezza blacklist sopra introdotto richiede un'interazione con l'utente, perciò si ritiene necessario introdurre un caso d'uso aggiuntivo "Gestione Blacklist" che si occuperà dell' aggiunta/rimozione degli utenti dalla blacklist.


SmartKey
<<Evento>>
Primo Avvio
Creazione Spazio Dedicato
Scelta di Cosa Sincronizzare
<<Extends>>
<<Extends>>
Gestione SmartKey
<<Extends>>
Gestione Blacklist
<<Uses>>
Controllo degli Accessi (Security)
<<Extends>>
<<Extends>>
<<Uses>>
Utente
Analisi e Ottimizzazione
Sincronizzazione
Sistema di identificazione
Uso Spazio Privato
Uso Spazio Pubblico
Blacklist (Security)
Modifica File Non propri
File Injection
Utente Distratto
Utente Malevolo

Scenari aggiuntivi

Gestione Blacklist

Titolo	Gestione Blacklist
Descrizione	Si occupa di aggiungere, o rimuovere da una lista, utenti indesiderati per la sincronizzazione di file da loro creati in cartelle pubbliche altrui
Attori	Utente
Relazioni	Gestione SmartKey, Sincronizzazione
Precondizioni	• Il software SmartKey deve essere correttamente installato nel sistema e in funzione • Il dispositivo di archiviazione USB deve essere correttamente inserito nel PC
Postcondizioni	La BlackList è impostata correttamente secondo la volontà dell'utente proprietario
Scenario principale	1. Viene avviata la sincronizzazione da parte dell'utente (come da punto 4 dello scenario principale di sincronizzazione) 2. Per ogni file creato da un altro utente nella propria cartella pubblica viene proposta all'utente una finestra pop-up, contente la richiesta di aggiunta dell'altro utente alla propria blacklist. 3. L'utente può: a. aggiungere l'altro utente alla propria blacklist, così che ogni file creato da tale utente viene ignorato ed eliminato durante la sincronizzazione b. ignorare la richiesta e quindi accettare la sincronizzazione di tale file.
Scenari alternativi	1. L'utente apre il software SmartKey, raggiunge l'interfaccia di blacklist, e si trova davanti a una sottofinestra contenente una lista di utenti appartenenti alla blacklist dell'utente attuale. 2. Premendo l'apposito bottone l'utente può rimuovere dalla blaklist gli utenti selezionati
Requisiti non funzionali	
Punti aperti	

Security Use Case & Misuse Case Scenari

<table>
<tr><td>Titolo</td><td>Sistema di identificazione</td></tr>
<tr><td>Descrizione</td><td>Identifica l’utente</td></tr>
<tr><td>Misuse Case</td><td>Modifica file non propri</td></tr>
<tr><td>Precondizioni</td><td></td></tr>
<tr><td>Postcondizioni</td><td>L’utente è stato identificato.</td></tr>
<tr><td>Scenario principale</td><td>
<table>
<tr><th>Sistema</th><th>Attaccante</th></tr>
<tr><td></td><td>Un utente malevolo o distratto cerca di scoprire le giuste credenziali di accesso e di compiere azioni che non gli sono permesse.</td></tr>
<tr><td>Controlla le credenziali di accesso e blocca le azioni di conseguenza.</td><td></td></tr>
</table>
</td></tr>
<tr><td>Scenario di attacco con avvenuto successo</td><td>
<table>
<tr><th>Sistema</th><th>Attaccante</th></tr>
<tr><td></td><td>L’attaccante riesce a scoprire le credenziali di accesso e compie azioni che non gli sarebbero permesse.</td></tr>
<tr><td>Verificate le credenziali di accesso permette l’esecuzione di tutte le azioni richieste dall’attaccante</td><td></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Titolo</td><td>Controllo degli accessi</td></tr>
<tr><td>Descrizione</td><td>L’accesso ai file sul dispositivo USB devono essere controllati</td></tr>
<tr><td>Misuse Case</td><td>Modifica file non propri</td></tr>
<tr><td>Precondizioni</td><td>L’attaccante o un qualunque utente entrano in possesso del dispositivo USB.</td></tr>
<tr><td>Postcondizioni</td><td>Il sistema, grazie alle ACL, nega l’operazione di modifica del file.</td></tr>
<tr><td>Scenario principale</td><td>
<table>
<tr><th>Sistema</th><th>Attaccante</th></tr>
<tr><td></td><td>Dopo essere entrati in possesso del dispositivo USB, l’attaccante cerca di modificare i file nella cartella privata dell’utente.</td></tr>
<tr><td>Controlla le regole presenti e blocca la modifica del file.</td><td></td></tr>
</table>
</td></tr>
<tr><td>Scenario di attacco con avvenuto successo</td><td>
<table>
<tr><th>Sistema</th><th>Attaccante</th></tr>
<tr><td></td><td>L’attaccante riesce a eludere le regole delle ACL.</td></tr>
<tr><td>Il sistema consente la modifica del file</td><td></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Titolo</td><td>Blacklist</td></tr>
<tr><td>Descrizione</td><td>La sincronizzazione da cartella pubblica a cartella privata deve essere controllata</td></tr>
<tr><td>Misuse Case</td><td>File Injection</td></tr>
<tr><td>Precondizioni</td><td>L’attaccante o un qualunque utente entrano in possesso del dispositivo USB.</td></tr>
<tr><td>Postcondizioni</td><td>L’utente ha effettuato una scelta(sincronizzare o eliminare) riguardante il file da sincronizzare.</td></tr>
<tr><td>Scenario principale</td><td>
<table>
<tr><th>Sistema</th><th>Attaccante</th></tr>
<tr><td></td><td>Dopo essere entrati in possesso del dispositivo USB, l’attaccante inserisce o modifica un file all’interno della cartella pubblica.</td></tr>
<tr><td>Inizia la sincronizzazione e chiede all’utente se si fida dell’autore di quel file.
In caso positivo, il file viene sincronizzato, in caso negativo il file viene eliminato.</td><td></td></tr>
</table>
</td></tr>
</table>

Requisiti di Protezione dei Dati

Dall’analisi del rischio si evincono ulteriori requisiti:

1. La creazione di vari log per tenere traccia di tutte le azioni compiute sul sistema.
2. Usare un sistema di identificazione che consente di utilizzare le ACL e individuarne una configurazione corretta
3. La creazione e gestione di una blacklist,che consenta all’utente di inserire gli utenti(da lui non riconosciuti) che vogliono sincronizzare dei contenuti dalla sua cartella pubblica alla sua privata.

Analisi del problema

Analisi Documento dei Requisiti: Analisi delle Funzionalità

Tabella Funzionalità

Funzionalità	Tipo	Grado Complessità
Gestione SmartKey	Memorizzazione dati, gestione dati	complessa
Protezione cartelle utenti	Protezione dati	semplice
Scrittura log	Memorizzazione dati, gestione dati	semplice

Gestione SmartKey: Tabella Informazioni/Flusso

Informazione	Tipo	Livello protezione / privacy	Input/Output	Vincoli
Path Sorgente	semplice	Protezione media	Input	Non più di 260 caratteri
Path Destinazione	semplice	Protezione media	Input	Non più di 260 caratteri
Account	composto	Protezione alta	Input	
Data	semplice	Protezione media	Output	
Ora	semplice	Protezione media	Output	
Operazione	composto	Protezione alta	Output	
Consiglio di ottimizzazione	semplice	Protezione bassa	Output	
Cartelle da comprimere	composto	Protezione media	Input	
Cartelle compresse	composto	Protezione media	Output	
Nome Utente	semplice	Protezione	Input	

		media		
Nome Dispositivo	semplice	Protezione media	Input	Non più di 256 caratteri

Protezione Cartella Utenti: Tabella Informazioni/Flusso

Informazione	Tipo	Livello protezione / privacy	Input/Output	Vincoli
Cartella Privata	semplice	Protezione alta	Input	Massimo 260 caratteri
Utente	composto	Protezione alta	Input	Massimo 256 caratteri
Blacklist	composto	Protezione media	Output	
Impostazioni Access Control List	composto	Protezione media	Input	

Scrittura Log: Tabella informazioni/ Flusso

Informazione	Tipo	Livello protezione / privacy	Input/Output	Vincoli
Data	semplice	Protezione media	Input	Non più di 40 caratteri
Ora	semplice	Protezione media	Input	Non più di 40 caratteri
Operazione Eseguita	composto	Protezione alta	Input	
Riga Log	composto	Protezione alta	Output	

Analisi Documento dei Requisiti: Analisi dei Vincoli

Tabella Vincoli

Requisito	Categorie	Impatto	Funzionalità
Trasparenza operazioni	Usabilità	Cercare di migliorare	Gestione SmartKey
Interfacce intuitive	Usabilità	Cercare di migliorare	Gestione SmartKey
Realizzazione in linguaggio C#	Organizzativo	Tecnologia	Tutte

Analisi Documento dei Requisiti: Analisi delle Interazioni

Tabella Maschere

Maschera	Informazioni	Funzionalità
Home SmartKey	Dati generali dei dispositivi USB	Gestione SmartKey
View Impostazioni Sincronizzazione	Path Sorgente, Path Destinazione, Verso, Account	Gestione SmartKey
View Analisi e Ottimizzazione	Consiglio di ottimizzazione, File da comprimere, File compressi, Memoria occupata, Nome utente, Nome dispositivo	Gestione SmartKey
View Blacklist	Blacklist	Gestione SmartKey
View Log	Riga log	Visualizzazione log

Tabella Sistemi Esterni

Sistema	Descrizione	Protocollo di interazione	Livello di Protezione
Access Control List (ACL)	Sistema che si occupa della sicurezza dell'accesso alle risorse	Mette a disposizione una serie di configurazioni per definire i permessi che un utente qualsiasi ha nei confronti di file e cartelle	Alto livello di sicurezza perché lavora a livello di sistema operativo
Esplora Risorse Windows	Sistema che si occupa della gestione di un filesystem	Rende disponibile un'interfaccia grafica per la visualizzazione e modifica di file in un filesystem	Basso livello di sicurezza essendo accessibile da chiunque

Analisi Ruoli e Responsabilità

Tabella Ruoli

Ruolo	Responsabilità	Maschere	Riservatezza	Numerosità
Utente	Gestione della propria cartella	Home SmartKey, View Impostazioni Sincronizzazione, View Analisi e Ottimizzazione, View Blacklist, View Log	È richiesto un alto grado di riservatezza	Non definibile a priori, strettamente dipendente dalla memoria disponibile nel dispositivo

Utente: Tabella Ruolo-Informazioni

Informazione	Tipo di Accesso
Dati Generali dei dispositivi USB	Lettura
Path Sorgente	Lettura/scrittura
Path Destinazione	Lettura/scrittura
Account	Lettura
Consiglio di ottimizzazione	Lettura
Cartelle da comprimere	Lettura/scrittura
Cartelle compresse	Lettura
Nome utente	Lettura
Nome dispositivo	Lettura
Blacklist	Lettura/scrittura
Riga log	Lettura
Cartella Privata	Lettura/scrittura

Scomposizione del problema

Tabella Scomposizione Funzionalità

Funzionalità	Scomposizione
Gestione Smartkey	Sincronizzazione, Scelta di cosa sincronizzare, Analisi e Ottimizzazione**
Analisi e Ottimizzazione	Analisi compressione, Calcolo consiglio, Compressione

**La funzione “Analisi e Ottimizzazione” è a sua volta scomposta nella riga sotto

Gestione Smartkey: Tabella Sotto-funzionalità

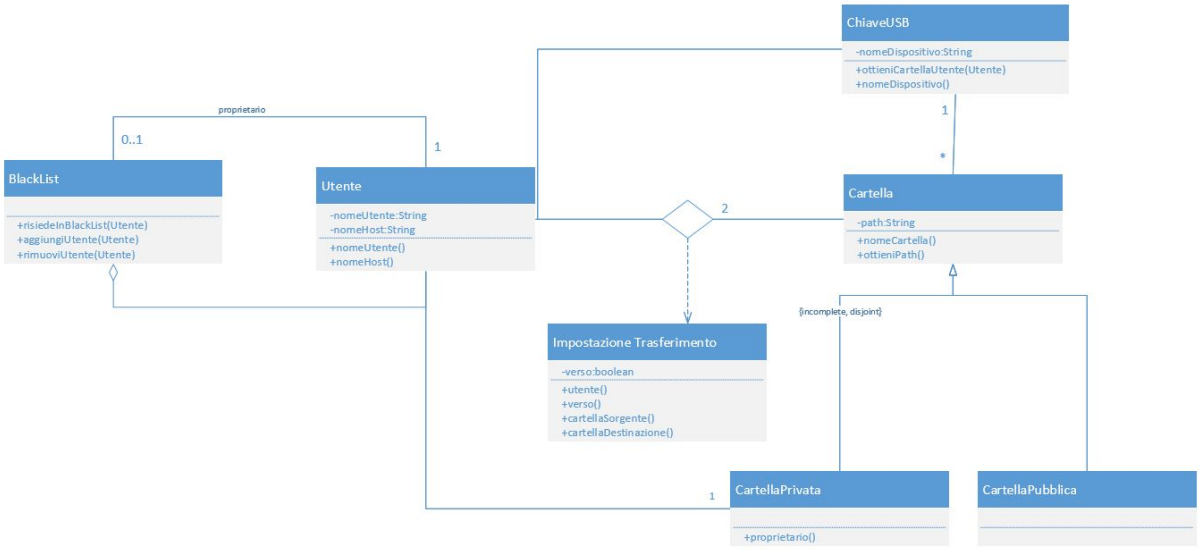
Sotto-Funzionalità	Sotto-Funzionalità	Legame	Informazioni
Sincronizzazione	Scelta di cosa sincronizzare	La sincronizzazione si basa sulla scelta di cosa sincronizzare	Utente, Path Sorgente, Path Destinazione

Gestione Analisi e Ottimizzazione: Tabella Sotto-funzionalità

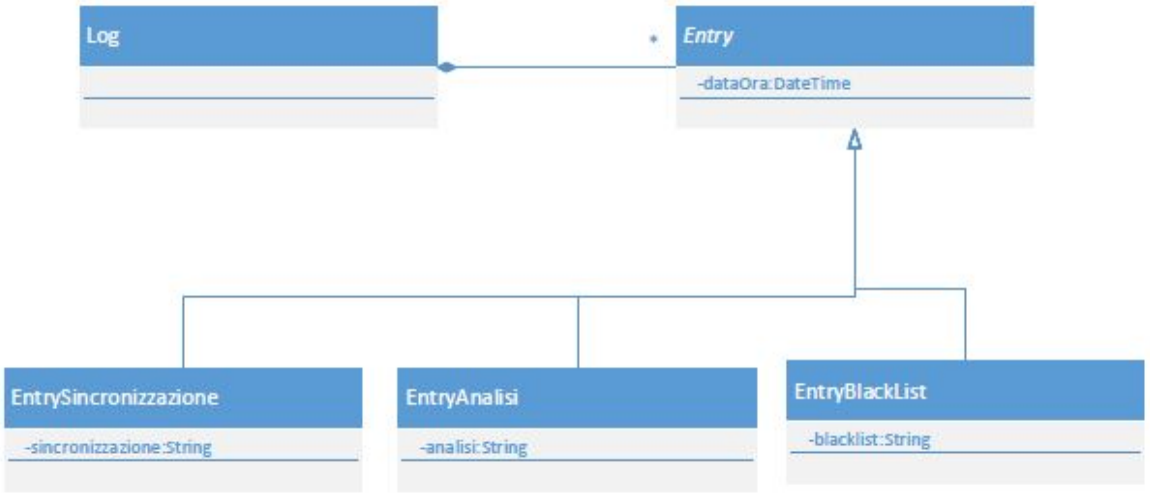
Sotto-Funzionalità	Sotto-Funzionalità	Legame	Informazioni
Analisi compressione	Calcolo consiglio	Il calcolo del consiglio verrà fatto in base all’analisi compressione	Consiglio Ottimizzazione,
Compressione	Calcolo Consiglio	La compressione della/e cartella/e potrà essere eseguita solamente se viene dato un consiglio positivo	Consiglio Ottimizzazione.

Creazione Modello del Dominio

Il seguente diagramma delle classi rappresenta la parte di modello del dominio relativa alla gestione USB



Il seguente diagramma delle classi rappresenta la parte di modello del dominio relativa ai log



Architettura Logica: Struttura

Diagramma dei package

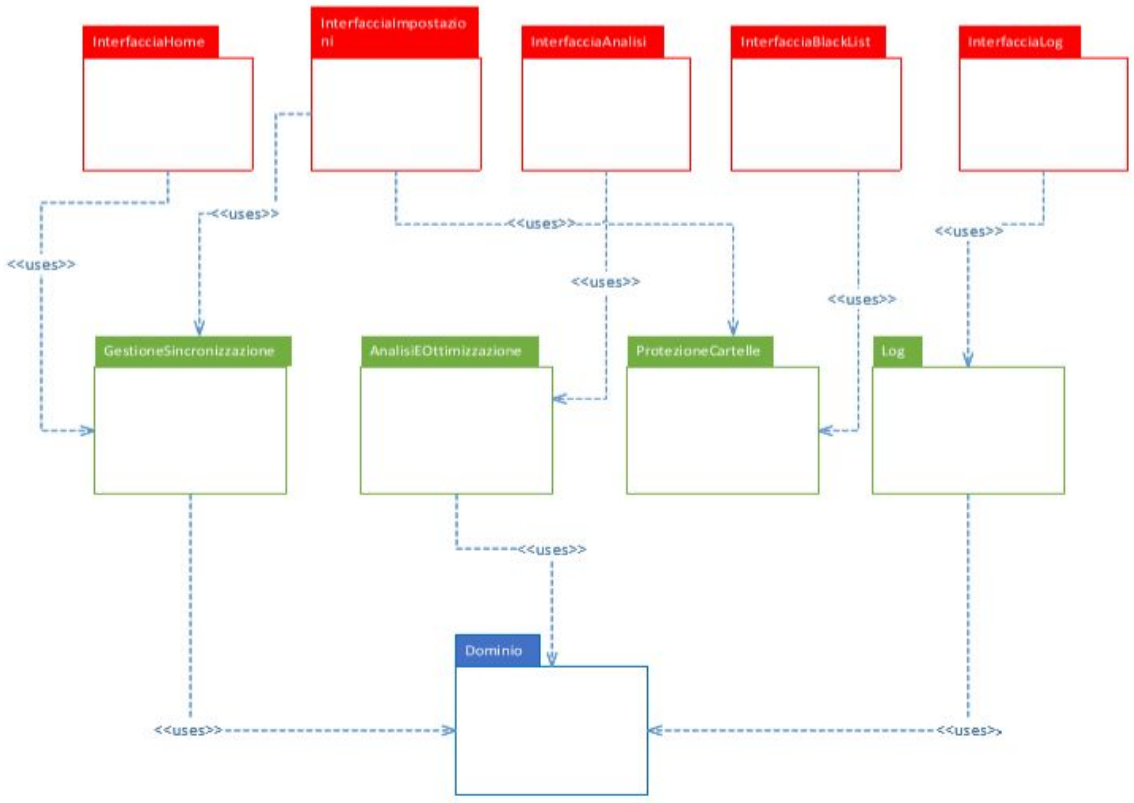


Diagramma delle classi: Dominio

Non viene riportato il diagramma delle classi associato al package Dominio in quanto è il modello del dominio creato nella fase precedente

Diagramma delle classi: InterfacciaBlackList & InterfacciaImpostazioni & InterfacciaHome

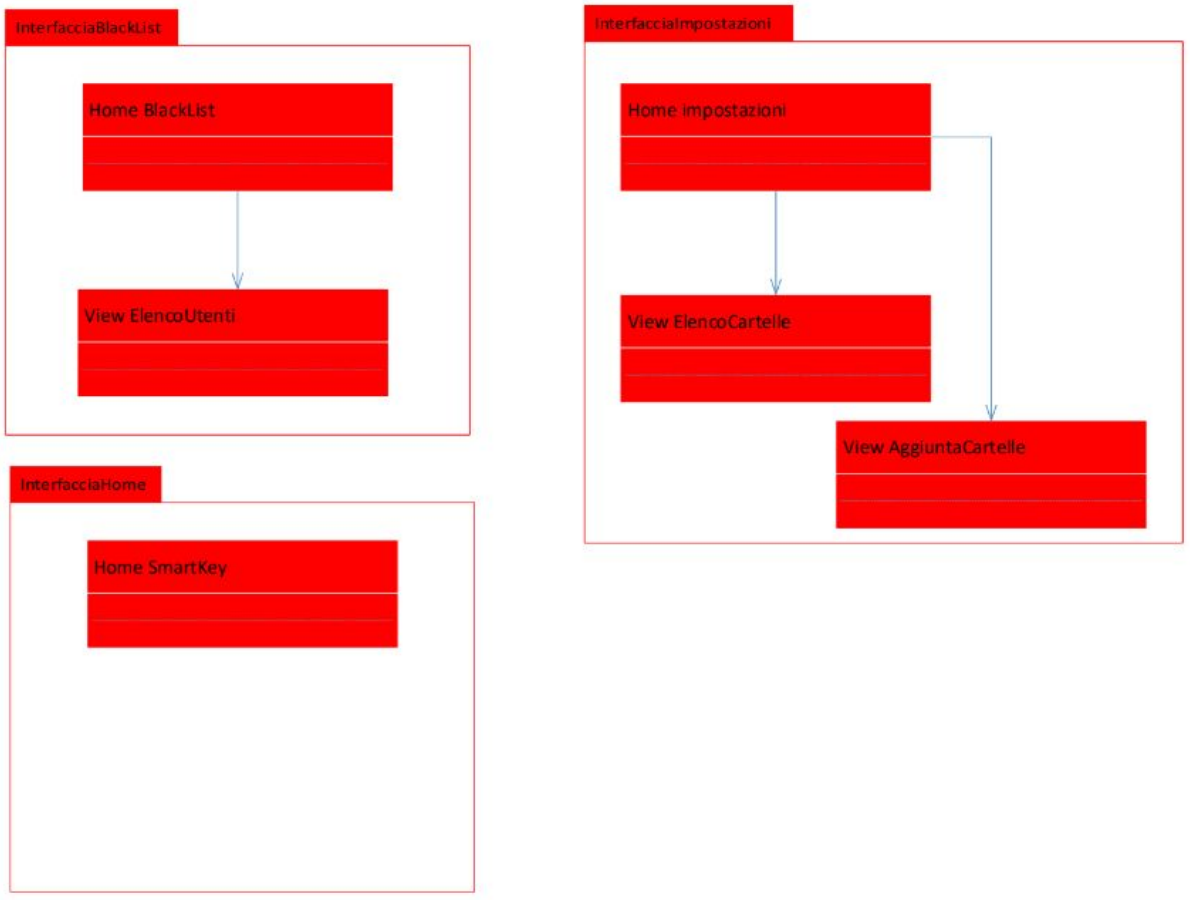


Diagramma delle classi: GestioneSincronizzazione

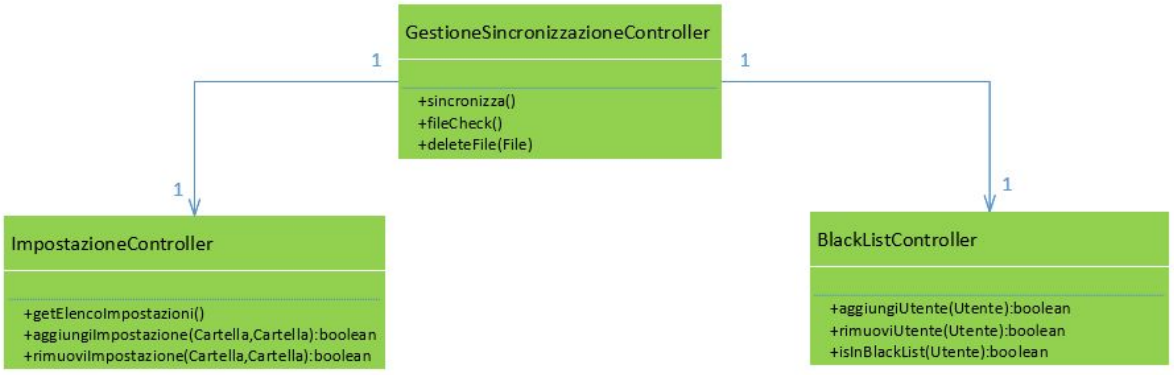


Diagramma delle classi: InterefacciaAnalisi

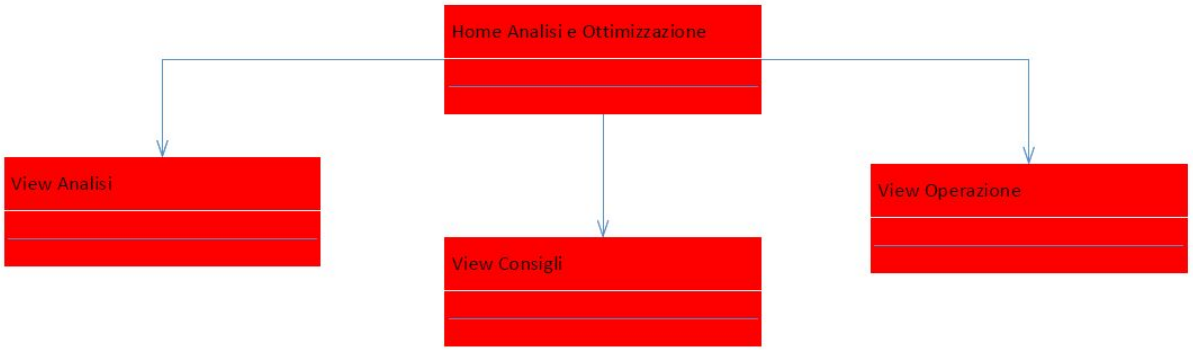


Diagramma delle classi: AnalisiEOttimizzazione

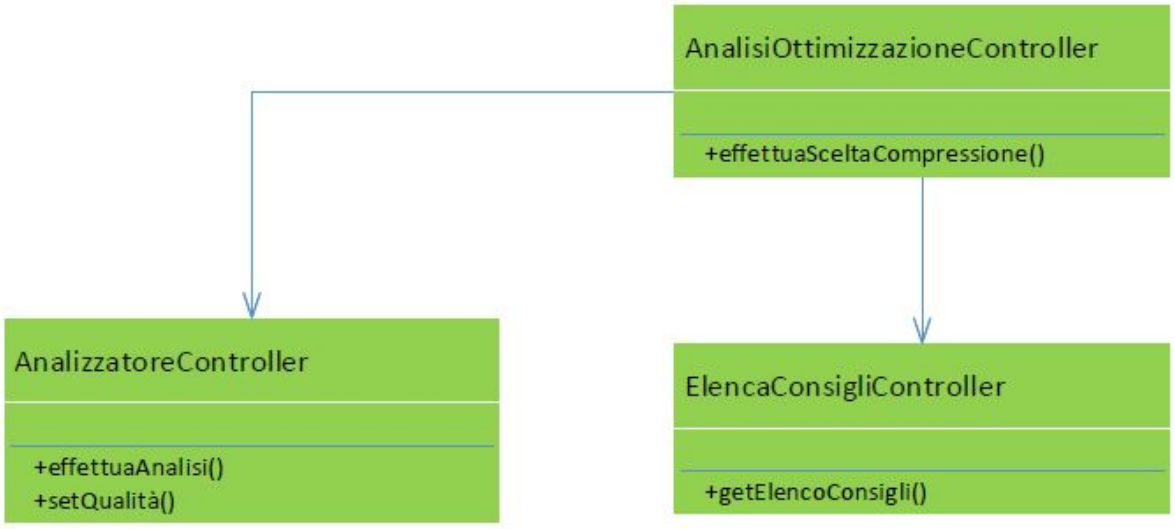
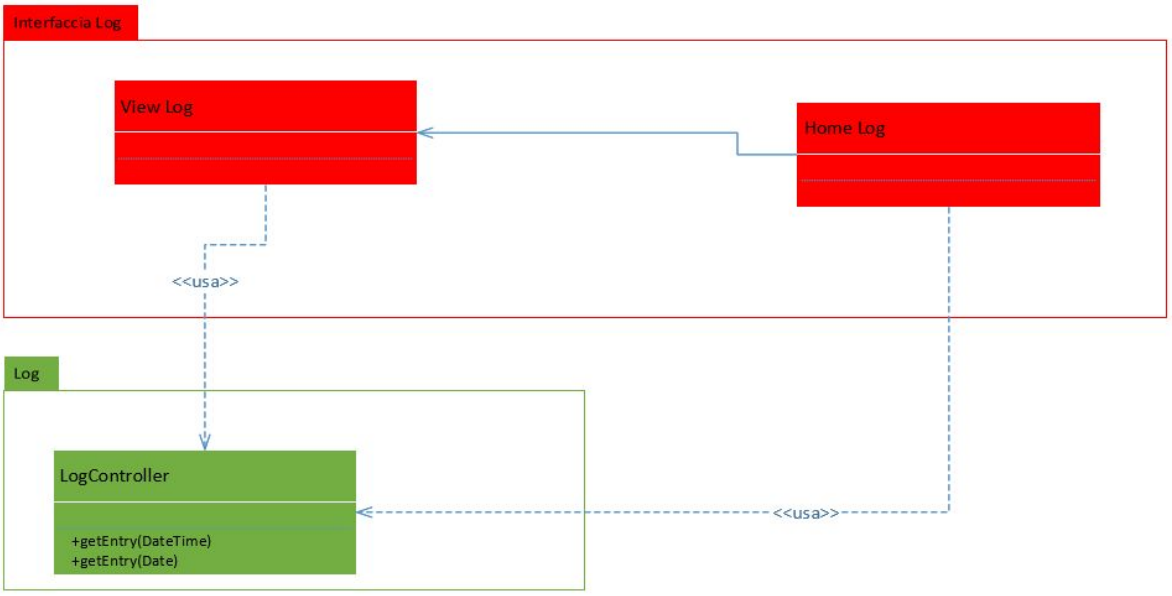


Diagramma delle classi: ProtezioneCartelle

Diagramma delle classi: InterfacciaLog & Log



Architettura Logica: Interazione

Diagramma di sequenza: AggiuntaBlackList

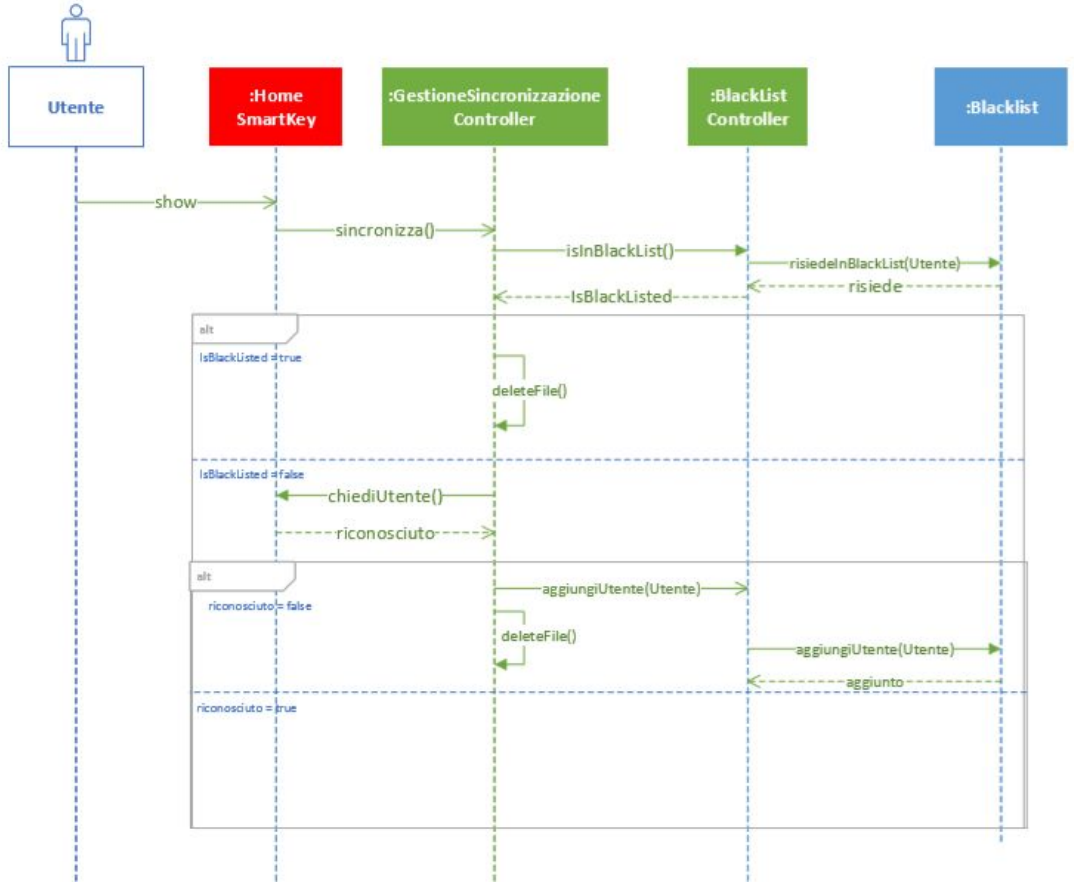
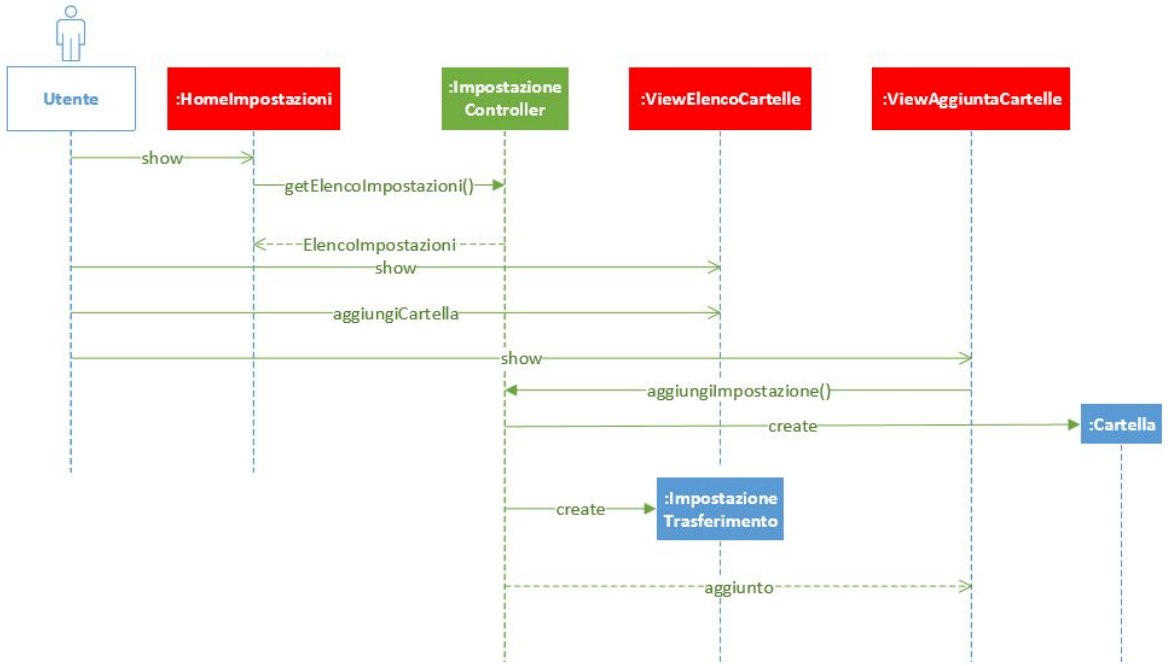


Diagramma di sequenza: AggiuntaImpostazione



Architettura Logica: Comportamento

Diagramma delle attività: Sincronizzazione Cartella

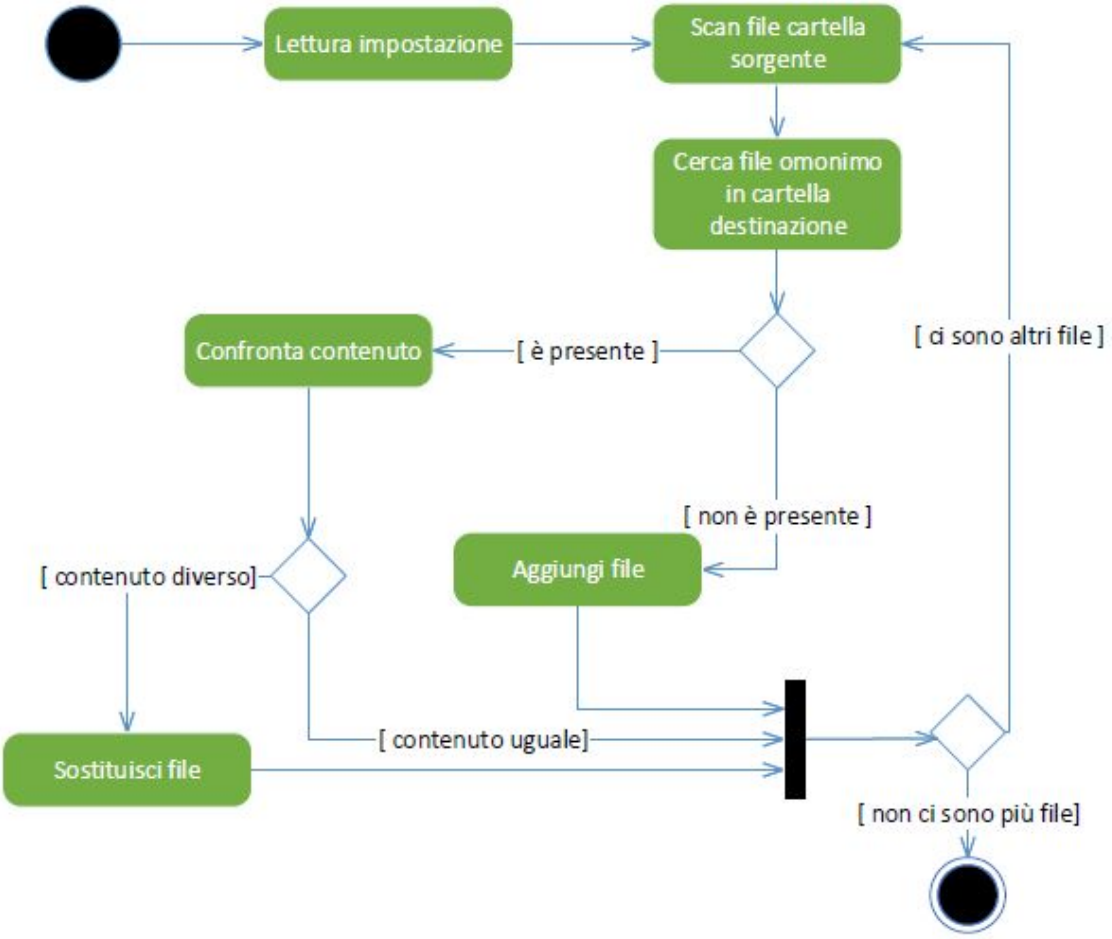
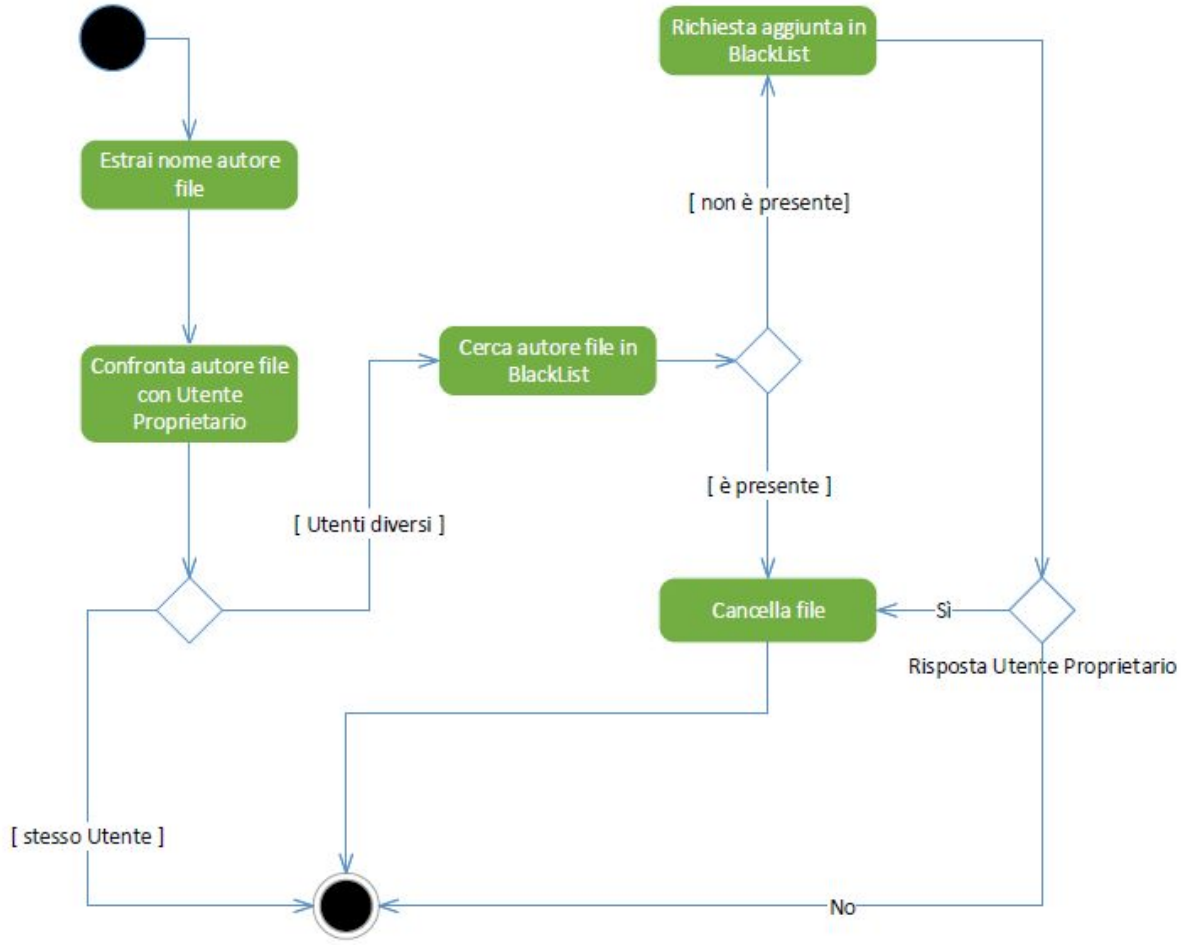


Diagramma delle attività: Scan file cartella sorgente



Piano di Lavoro

Package	Progetto	Sviluppo
Dominio	Arianna, Massimo, Riccardo	Massimo
GestioneSincronizzazione	Arianna, Massimo, Riccardo	Arianna, Massimo, Riccardo
AnalisiEOttimizzazione	Arianna, Massimo, Riccardo	Riccardo
ProtezioneCartelle	Arianna, Massimo, Riccardo	Massimo, Riccardo
Log	Arianna, Massimo, Riccardo	Arianna
InterfacciaHome	Arianna, Massimo, Riccardo	Arianna, Massimo, Riccardo
InterfacciaImpostazioni	Arianna, Massimo, Riccardo	Arianna
InterfacciaAnalisi	Riccardo	Riccardo
InterfacciaBlacklist	Massimo	Massimo
InterfacciaLog	Arianna	Arianna, Riccardo

Il prototipo da consegnare dovrà realizzare le funzionalità di: Sincronizzazione e Blacklist con la funzione di Log a loro associata e ovviamente anche le impostazioni di trasferimento dovranno essere funzionanti.
Con la persistenza delle impostazioni di trasferimento e della blacklist.

Tempi di rilascio
• Progettazione entro 20 giorni dal 1 Giugno 2018
• Sviluppo delle singole parti con collaudo unitario entro 10 giorni rispetto al fine della progettazione
• Integrazione e test dell'intero sistema entro 2 settimane rispetto alla fine dello sviluppo

Sviluppi futuri:
Il committente ha richiesto che il sistema in futuro abbia l'integrazione verso più sistemi di persistenza,si richiede al team di sviluppo di tenere conto di questo possibile sviluppo futuro.

Piano del Collaudo

A titolo di esempio si riportano i test della classe Cartella e della classe ImpostazioneTrasferimento.
Si sono testati i vincoli sull'input, la stringa di test è lunga 300 caratteri, nel caso che venga che il path superi il vincolo esplicitato(260 caratteri) verrà lanciata un'eccezione(PathNotValidException).

Test Cartella:

```
[TestFixture]
class CartellaTest
{
    //Questa stringa effettua il test sugli input
    //Se passata al costruttore, dovrebbe lanciare eccezione
    private readonly string _veryLongPath = "aaaaaaaaaaaaaaaaaaaa" +
        "aaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
        "aaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
        "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
        "aaaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
        "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
        "aaaaaaaaaaaaaaaaaaaaaaaaa" +
        "aaaaaaaaaaaaaaaaaaaaaaaaaaa" +
        "aaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
        "aaaaaaaaaaaaaaaaaaaaaa";


    //Test del costruttore dell'oggetto cartella
    [TestCase]
    public void TestCartellaConstructor()
    {
        Cartella c = new Cartella("ABC");
        Assert.AreNotEqual(c, null);
        Assert.AreEqual(c.Path, "ABC");
    }
    //Test che controlla che se il path supera 260 caratteri viene lanciata eccezione
    [TestCase]
    public void TestCartellaPathLength()
    {
        Cartella c = new Cartella("ABC");
        Assert.Throws<PathNotValidException>(() => c.Path = _veryLongPath);

    }

    //Test che controlla che la funzione removechild funzioni correttamente
    //e rimuova l'elemento specificato dalla lista
    [TestCase]
    public void TestCartellaRemoveChild()
    {
        Cartella home = new Cartella("ABC");
        Cartella subHome = new Cartella("KDE");
        home.AddChild(subHome);
        FilesystemElement toTest = home.GetChild("KDE");
        Assert.AreSame(subHome, toTest);
        home.RemoveChild(toTest);
        toTest = home.GetChild("KDE");
        Assert.That(toTest == null);
    }
```

```
//Test che controlla che venga aggiunto l'elemento specificato alla
//Lista di figli di quella cartella.
[TestCase]
public void TestCartellaAddChild()
{
    Cartella home = new Cartella("ABC");
    Cartella subHome = new Cartella("KDE");
    home.AddChild(subHome);
    FilesystemElement toTest = home.GetChild("KDE");
    Assert.AreSame(subHome, toTest);
}
```

Test ImpostazioneTrasferimento:

```
[TestFixture]
class ImpostazioneTrasferimentoTest
{
    //_veryLongElement è di 300 caratteri
    private readonly string _veryLongElement = "aaaaaaaaaaaaaaaaaaaa" +
      "aaaaaaaaaaaaaaaaaaaaaaaaaaa" +
      "aaaaaaaaaaaaaaaaaaaaaaaaaaa" +
      "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
      "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
      "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
      "aaaaaaaaaaaaaaaaaaaaaaaaa" +
      "aaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
      "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
      "aaaaaaaaaaaaaaaaaaaaaaaaa";

    [TestCase]
    public void TestImpostazioneThrowNullConstructor()
    {
        Assert.Throws<PathNotValidException>(() => new ImpostazioneTrasferimento("", null));
        Assert.Throws<PathNotValidException>(() => new ImpostazioneTrasferimento(null, ""));
        Assert.Throws<PathNotValidException>(() => new ImpostazioneTrasferimento(null, null));
    }
    [TestCase]
    public void TestImpostazioneThrowLengthConstructor()
    {
        Assert.Throws<PathNotValidException>(() => new ImpostazioneTrasferimento(_veryLongElement, ""));
        Assert.Throws<PathNotValidException>(() => new ImpostazioneTrasferimento("", _veryLongElement));
        Assert.Throws<PathNotValidException>(() => new ImpostazioneTrasferimento(_veryLongElement, _veryLongElement));
    }
    [TestCase]
    public void TestImpostazioneThrowNullSetter()
    {
        ImpostazioneTrasferimento impostazione = new ImpostazioneTrasferimento("folderS", "folderD");
        Assert.Throws<PathNotValidException>(() => impostazione.CartellaDestinazione = null);
        Assert.Throws<PathNotValidException>(() => impostazione.CartellaSorgente = null);
    }
    [TestCase]
    public void TestImpostazioneThrowLengthSetter()
    {
        ImpostazioneTrasferimento impostazione = new ImpostazioneTrasferimento("folderS", "folderD");
        Assert.Throws<PathNotValidException>(() => impostazione.CartellaDestinazione = _veryLongElement);
        Assert.Throws<PathNotValidException>(() => impostazione.CartellaSorgente = _veryLongElement);
    }
```

```
    [TestCase]
    public void TestImpostazioneSetter()
    {
        ImpostazioneTrasferimento impostazione = new ImpostazioneTrasferimento("myDir", "yourDir");
        impostazione.CartellaDestinazione = "CambiamentoDestinazione";
        impostazione.CartellaSorgente = "CambiamentoSorgente";
        Assert.AreEqual(impostazione.CartellaDestinazione, "CambiamentoDestinazione");
        Assert.AreEqual(impostazione.CartellaSorgente, "CambiamentoSorgente");
    }

    [TestCase]
    public void TestImpostazioneGetter()
    {
        ImpostazioneTrasferimento impostazione = new ImpostazioneTrasferimento("myDir", "yourDir");
        Assert.AreEqual(impostazione.CartellaSorgente, "myDir");
        Assert.AreEqual(impostazione.CartellaDestinazione, "yourDir");
    }
```

Progettazione

Progettazione Architetturale

Requisiti Non Funzionali

Nella tabella dei vincoli (Analisi del problema) sono emersi due tipi di requisiti non funzionali richiesti dal sistema:

- Usabilità
- Organizzativo

L'usabilità richiesta nella fase di analisi del problema (trasparenza delle operazioni) incide molto con l'implementazione delle funzionalità, quindi con la tecnologia utilizzata ,poichè una soluzione poco efficiente potrebbe rendere le operazioni onerose per il sistema e l'esecuzione delle funzionalità non sarebbe più trasparente come richiesto.
Molte scelte inerenti la progettazione dovranno poi essere coerenti con gli strumenti richiesti dal cliente, come il linguaggio di programmazione C#.

Scelta dell'Architettura

Dal punto di vista architetturale, l'architettura più idonea per questo tipo di sistema è un'achitettura MVP (Model View Presenter) che è una variante di MVC (Model View Controller) che permette una migliore separazione tra view e logica di dati.
Model:
Il modello gestisce i dati e le entità in gioco quali:

- Utente
- Cartella
- BlackList
- Impostazioni
- Log
- Chiave USB

View:
La view mostra i dati e le azioni eseguite del Software Smartkey nelle maschere seguenti:

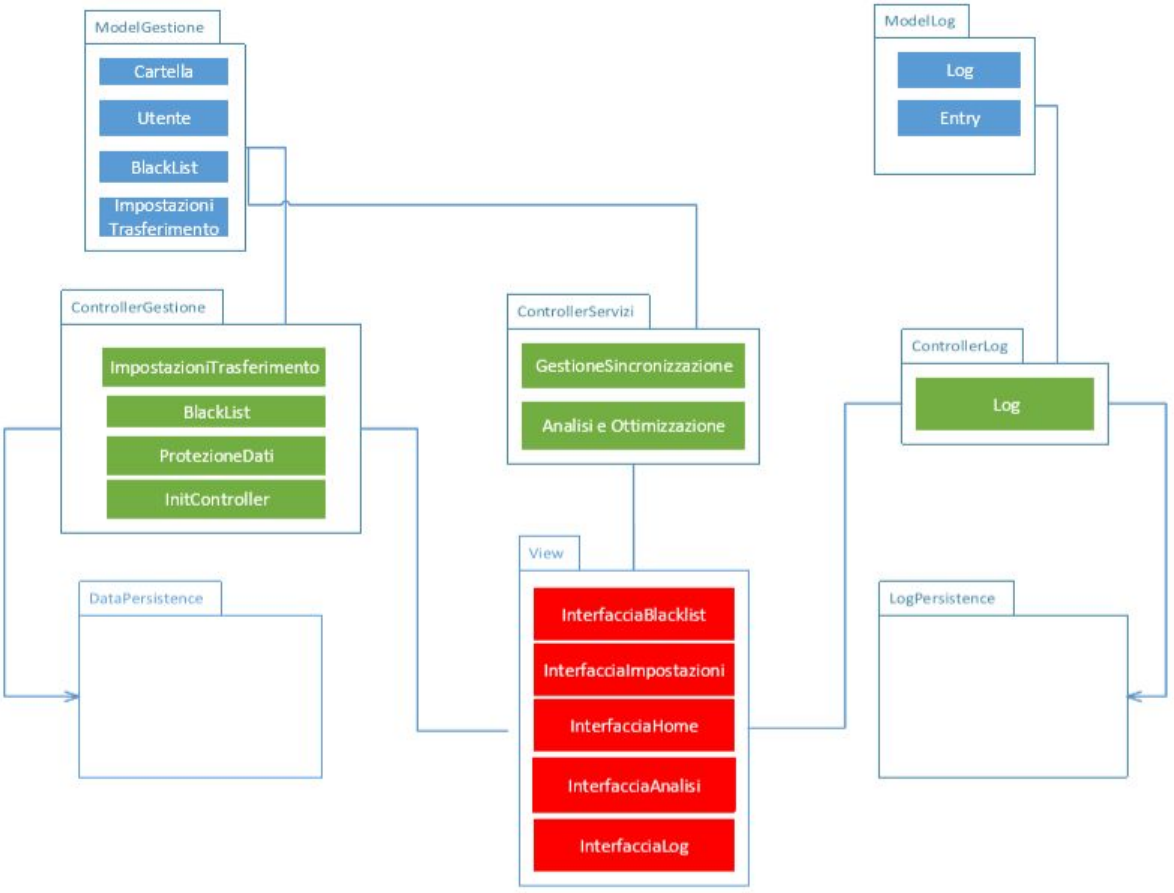
- Home SmartKey
- View ImpostazioniSincronizzazione
- View Analisi e Ottimizzazione
- View BlackList
- View Log

Presenter:
Il presenter manipola i dati per ottenere le funzionalità richieste quali:

- Gestione Sincronizzazione
- Scrittura Log
- Protezione dei Dati
- Analisi e Ottimizzazione

Si comporta come un controller di un'architettura MVC con la differenza che media tra Model e View senza farli comunicare e decide cosa succede quando si interagisce con la view.

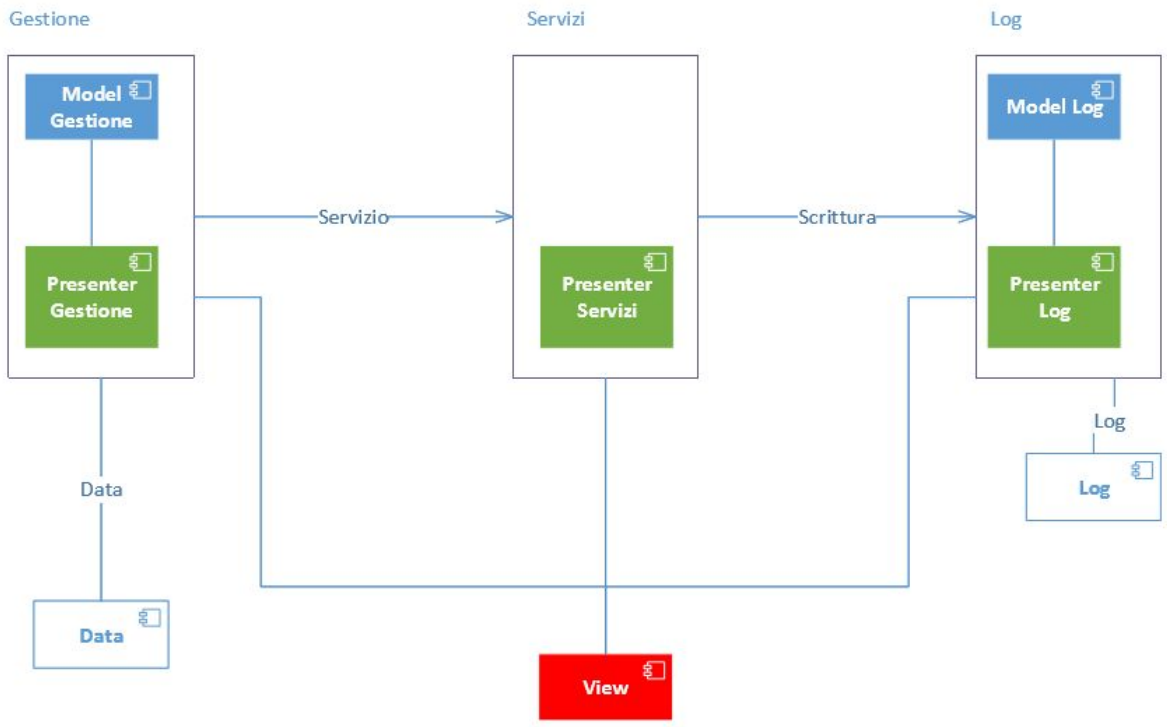
Nella figura sottostante è riportata l'Architettura del Sistema organizzata attraverso un diagramma dei package opportunamente ampliato rispetto al precedente stilato nell'analisi del problema.



Si è deciso di inserire un presenter che gestisce tutti gli altri presenter e si interfaccia con il supporto alla persistenza.

Nel package Model Log con Entry si intende, oltre che la classe astratta, anche tutte le sue concretizzazioni, le quali non sono state inserite per una migliore visibilità del diagramma.

Nella figura sottostante è riportata l'Architettura del Sistema organizzata attraverso un diagramma dei componenti, visivamente semplificato dai macro-componenti gestione, servizi e log, ciascuno contenente il proprio model e presenter.



Spiegata la decisione del pattern architetturale e mostrato il diagramma dei package inerente, per una più facile comprensione delle classi il progettista si riferirà ai presenter col termine ”controller”, mantenendo tuttavia la struttura MVP.

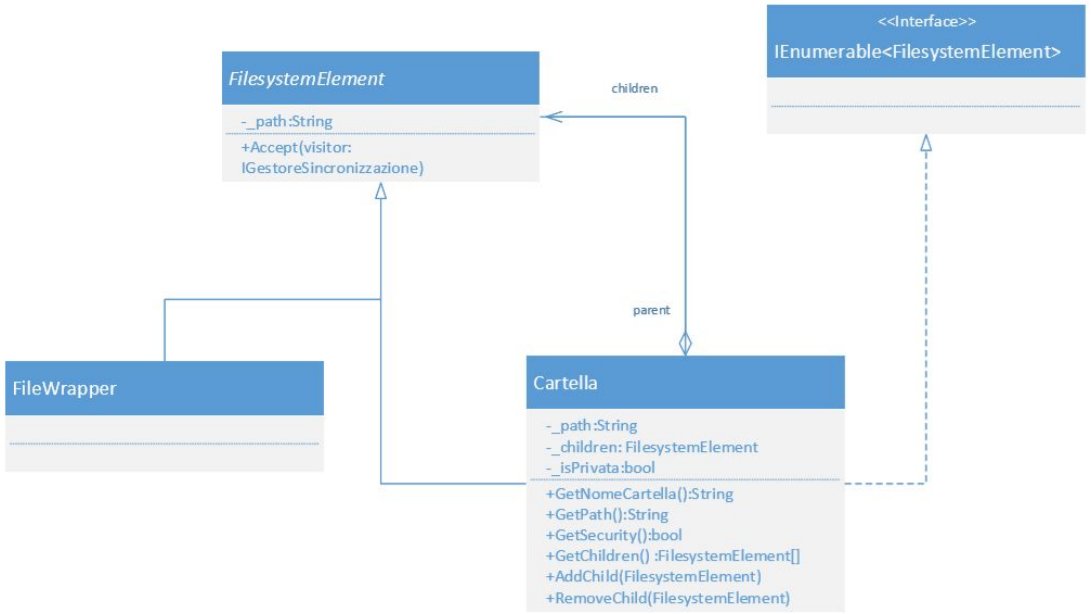
Progettazione di Dettaglio

Nel seguito si riportano i diagrammi di dettaglio delle varie parti del Sistema.

Struttura

I cambiamenti del dominio seguente rispetto a quello dell'analisi del problema sono dovuti alla decisione di applicare diversi Design Pattern ed il modello architetturale (MVP).Si è scelto inoltre di utilizzare il sistema di login di Windows come sistema di Identificazione, perciò l'identità di un utente qualsiasi verrà ricavata dal software SmartKey tramite Windows e memorizzatacome stringa.
Di conseguenza come sistema di ACL utilizzeremo quelle fornite dal Sistema Operativo Windows.

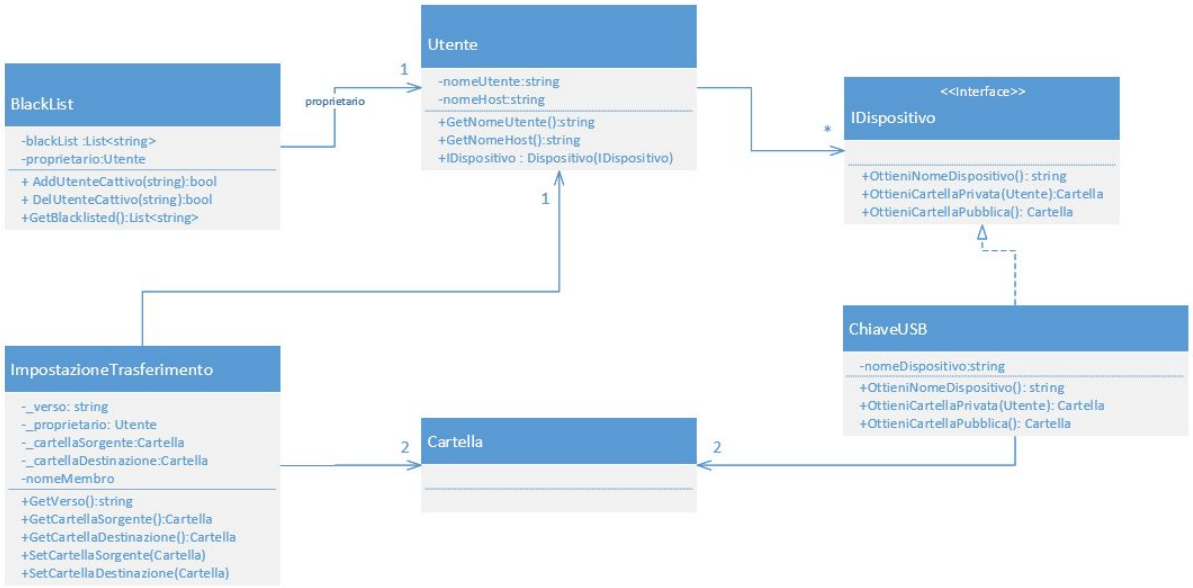
Diagramma di Dettaglio : Dominio-FileSystem



Data la natura gerarchica del filesystem che vogliamo rappresentare si è scelto di utilizzare il pattern Composite.
La classe astratta FilesystemElement ha il ruolo di "Component", ha anche un metodo Accept poiché, dato che su tutti gli elementi messi in gerarchia vogliamo effettuare delle operazioni (sincronizzazione) si è ritenuto opportuno separare le operazioni dal modello dei dati, quindi di applicare il pattern Visitor.
La classe FileWrapper rappresenta un qualsiasi file nonché la "Leaf" nel pattern Composite e la classe Cartella è l'elemento "Composite" del pattern.
Alla classe cartella è stato aggiunto il metodo GetSecurity che serve ad identificarla come cartella privata di un utente o meno.

Per poter avere una gestione degli errori migliore si è deciso di creare un'eccezione personalizzata quando un path supera i 260 caratteri come definito nell'analisi del problema o quando non ha valori significativi.

Diagramma di Dettaglio : Dominio-Blacklist-Utente-IDispositivo-ImpostazioneTrasferimento

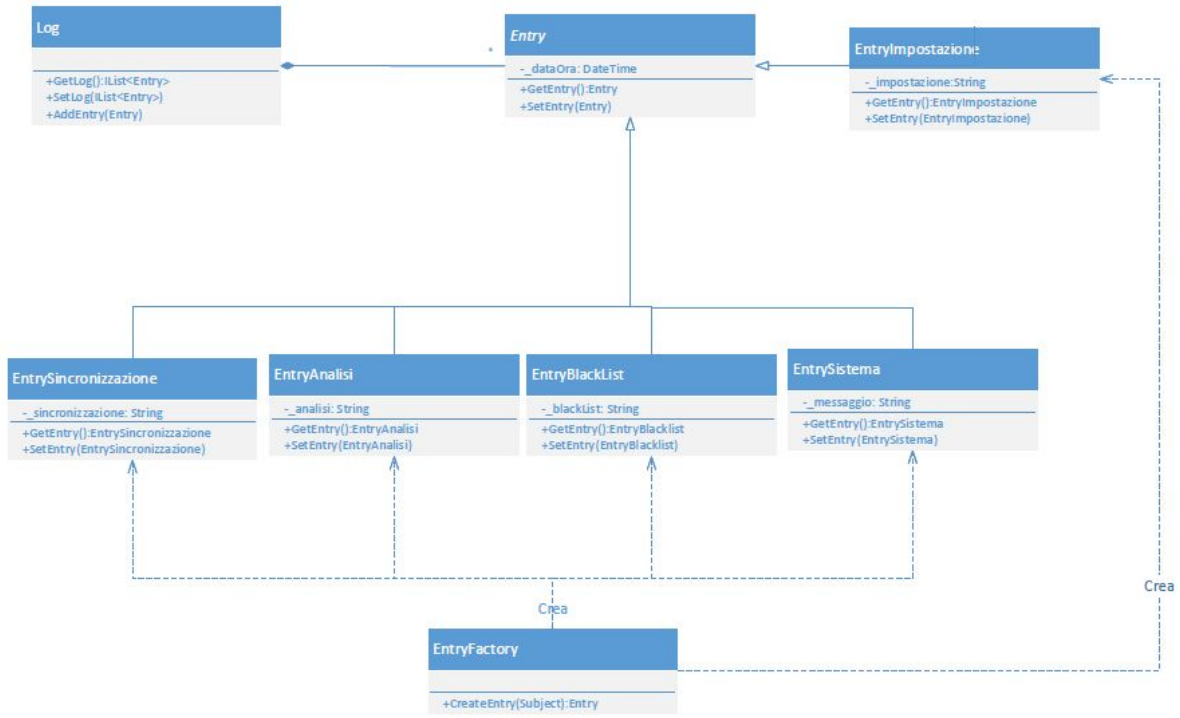


Rispetto al modello di dominio presentato nell'analisi del problema si è scelto di astrarre ChiaveUSB tramite l'interfaccia IDispositivo, in modo da garantire una facilità di estensione nel caso si vogliano utilizzare dispositivi differenti.

La relazione di aggregazione mostrata nell'analisi del problema tra utente e blacklist è stata semplificata, la blacklist avrà solamente una lista di stringhe(che rappresentano) gli utenti.

Per ImpostazioneTrasferimento si è deciso di aggiungere l'attributo "utente" e i metodi "SetCartellaSorgente" e "SetCartellaDestinazione" per maggiore precisione.

Diagramma di Dettaglio : Dominio-Log



Si è ritenuto opportuno aggiungere EntryImpostazione per monitorare al meglio anche questa attività svolta dal software SmartKey, sono stati inoltre aggiunti per ogni entry i rispettivi getters e setters.
Abbiamo inserito una classe EntryFactory che si occuperà di istanziare la entry giusta (Pattern Factory) in base all'istanza che notificherà l'observer.

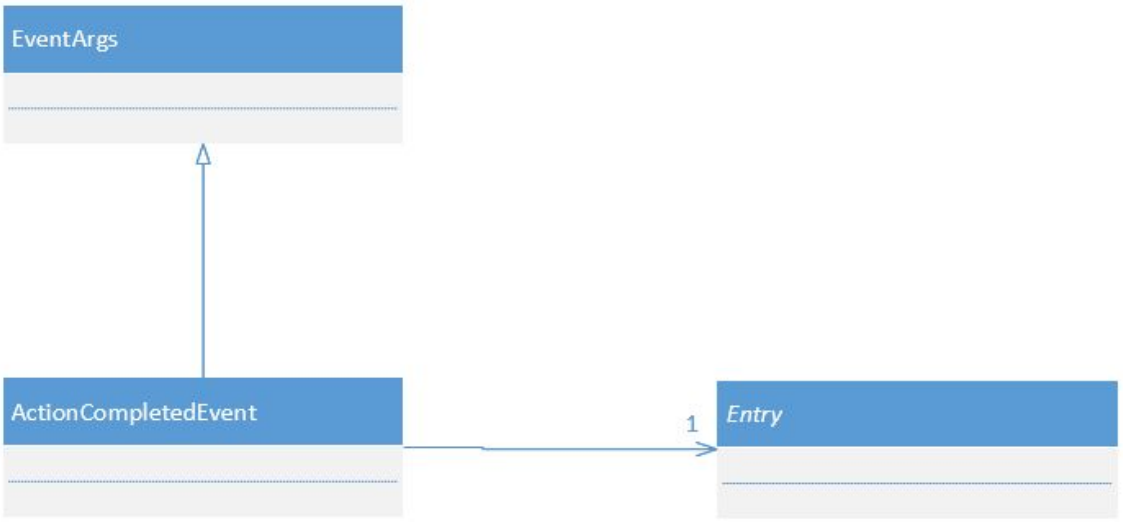
Diagramma di Dettaglio : Interfacce per i controller

Ogni interfaccia per i controller che realizza operazioni che devono essere inserite nel file di log dovrà estendere l'interfaccia IController che contiene solamente un evento (ToLog), utile per realizzare il pattern Observer.

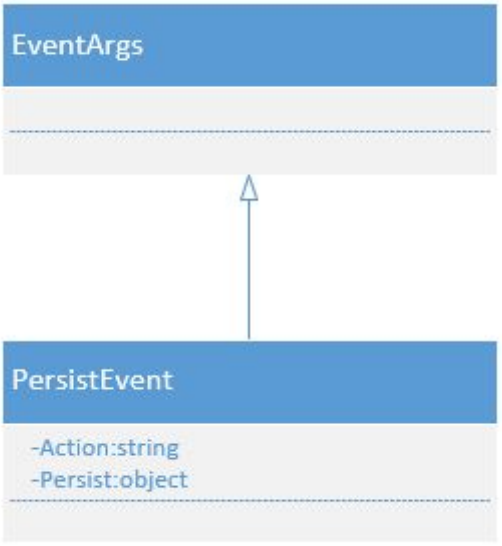
Anche le operazioni di persistenza saranno gestite a eventi, in particolare ogni volta che il controller aggiungerà o rimuoverà impostazioni di trasferimento o elementi nella blacklist dovrà scatenare l'evento ToPersist, in modo da rendere persistente la creazione/rimozione.

Un'istanza ActionCompletedEvent sarà passata come argomento quando verranno notificati gli handler registrati all'evento.
Questo evento conterrà appunto una entry da inserire nel log.

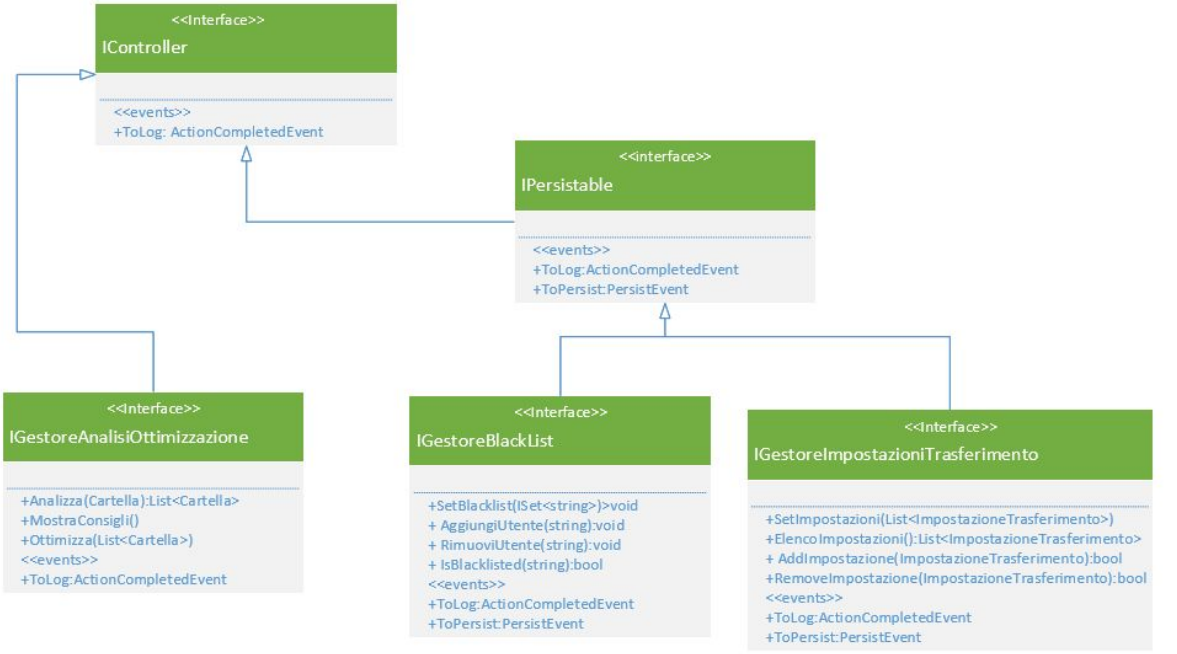
Questo è reso possibile grazie all'estensione della classe di sistema EventArgs.

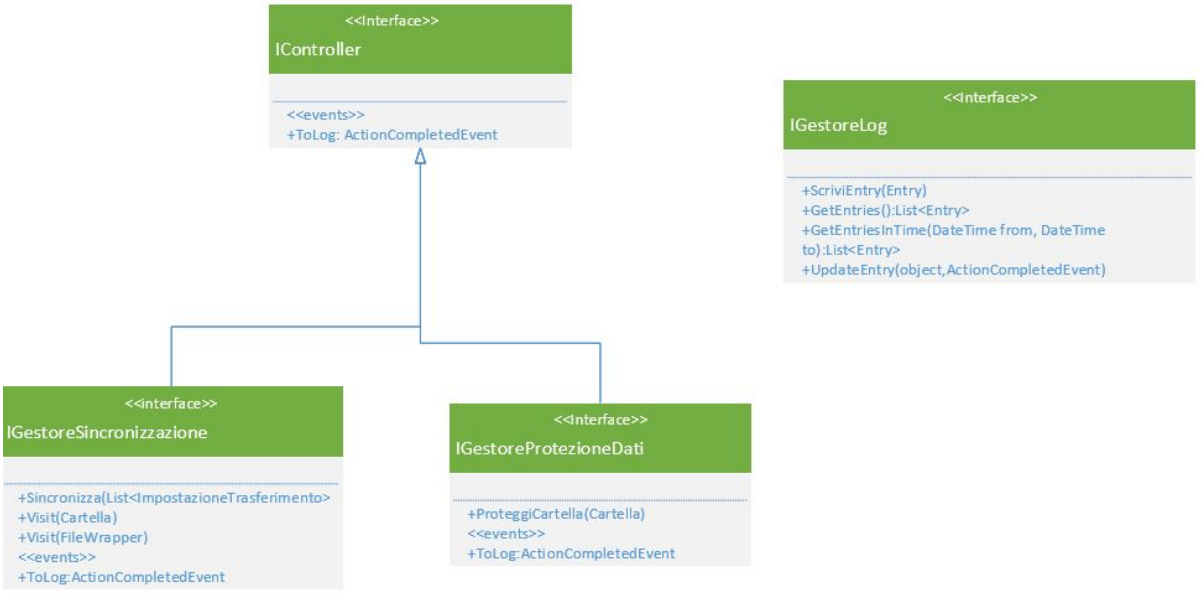


Invece per la persistenza è necessario passare come parametro il tipo di azione da effettuare(aggiungere/rimuovere) codificata come stringa.
E l'oggetto che vogliamo rendere persistente oppure che vogliamo rimuovere dal supporto alla persistenza.



Si è deciso di non mettere tutto in un'unica interfaccia per realizzare l'open/closed principle, nel quale per aggiungere funzionalità bisogna estendere un'entità, non modificarla.
Viene riportata la struttura delle interfacce dei controller.





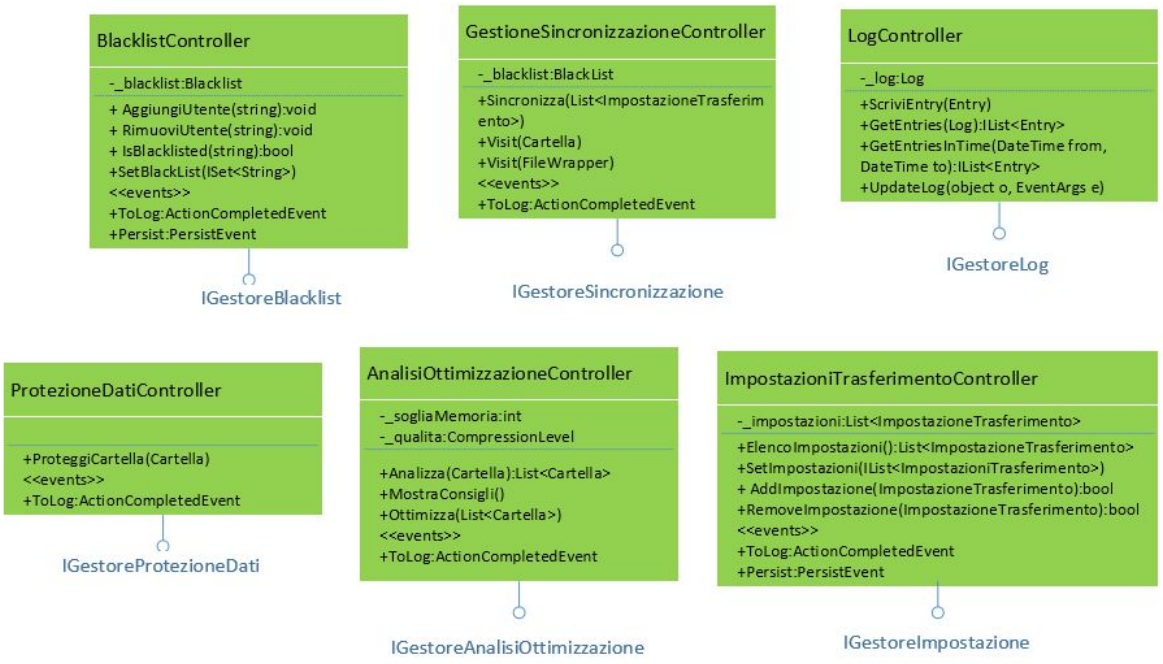
L'interfaccia "IGestoreSincronizzazione" è l'interfaccia che permette di realizzare il pattern visitor.

Diagramma di Dettaglio : Controller

Premessa:Tutti i metodi dell'analisi del problema sono stati modificati, a causa dell'applicazione dei vari design pattern e design principles.

Abbiamo scelto il pattern Observer per realizzare le funzioni di Log perchè in questo modo tutti i controller possono notificare un'operazione, che verrà scritta nel file di log.
Ogni controller si occuperà, in caso di un'azione, di creare la entry corretta grazie alla classe Factory EntryFactory.

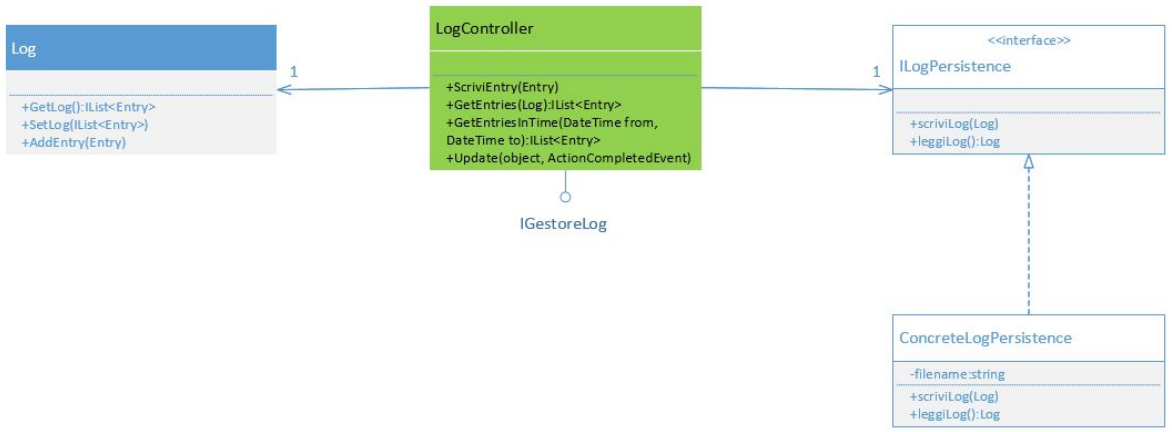
Di seguito sono mostrati i singoli controller:



- ProtezioneDatiController
    - Il metodo ProteggiCartella realizza le impostazioni di protezione accordate per l'utente, come descritte nell'analisi del rischio.
- AnalisiOttimizzazioneController
    - Rispetto all'analisi del problema si è scelto di raggruppare i 3 controller(AnalizzatoreController, ElencaConsigliController, CompressioneController) in unico controller,AnalisiEOttimizzazioneController.

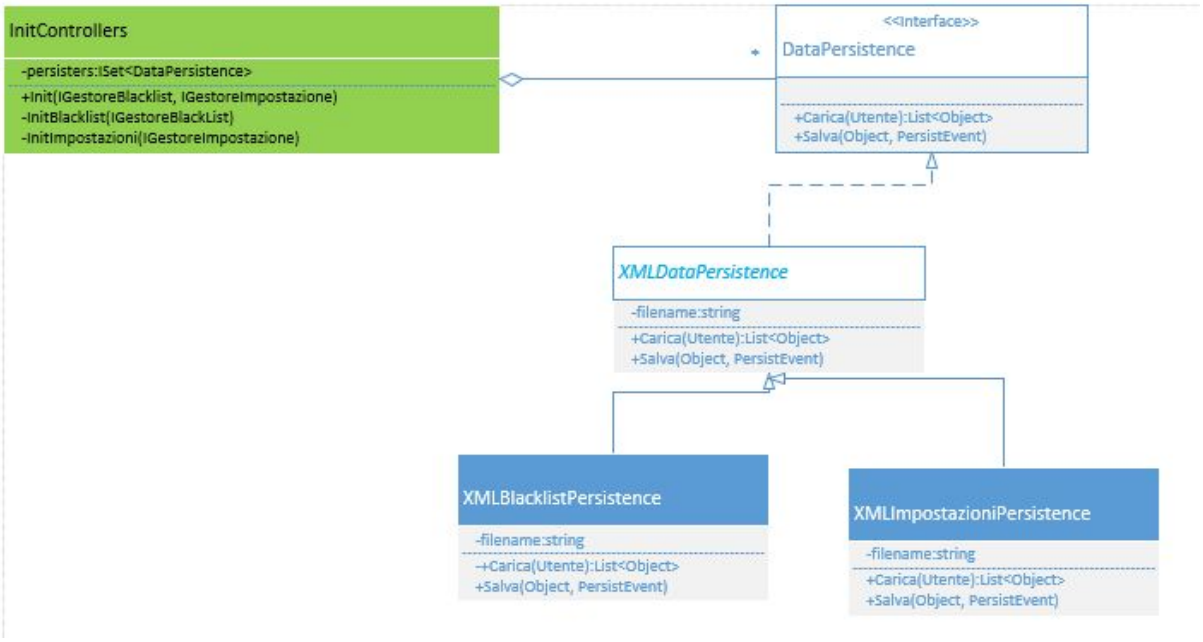
    - Le variabili sogliaMemoria e qualità servono rispettivamente per configurare l'analisi e l'ottimizzazione dei dati: sogliaMemoria indica il valore (in MB) sopra il quale è consigliabile comprimere una cartella, qualità è una variabile CompressionLevel che stabilisce il tipo di compressione da utilizzare.
    - Il metodo Analizza prende come parametro la cartella dell'utente e restituisce una lista di sotto-cartelle per le quali è consigliata la compressione.
    - Il metodo Ottimizza prende come parametro in ingresso una lista di cartelle designate alla compressione tra quelle consigliate
- ImpostazioniTrasferimentoController

    - La variabile impostazioni è una lista contenente le impostazioni di trasferimento dell'utente.
    - Sono presenti metodi per aggiungere e rimuovere singole impostazioni e per inserire una lista di impostazioni.
- BlacklistController
    - Vi sono metodi per aggiungere e rimuovere utenti dalla Blacklist,è possibile inserire una nuova blacklist,inoltre è presente il metodo IsBlackListed che verifica se un certo utente è presente nella lista.
- GestioneSincronizzazioneController
    - Il metodo Sincronizza prende in ingresso una lista di impostazioni ed esegue la sincronizzazione seguendone le direttive controllando la blacklist.
    - I metodi Visit (pattern Visitor) servono per ricorstruire il filesystem di nostro interesse per la sincronizzazione.
- LogController
    - Oltre ai metodi ScriviEntry e GetEntries è presente il metodo GetEntriesInTime che restituisce una collezione di Entry in un certo margine di tempo specificato dai parametri della funzione
    - Il metodo UpdateLog sarà l'handler quello che sarà agganciato agli eventi dei vari controller.



In questo schema viene rappresentato come LogController contenga un oggetto di tipo Log e per la persistenza del file di log usi un'interfaccia di tipo ILogPersistence.
Tutto il compito di scrittura su file del log, con i formati sotto riportati, sarà compito della classe ConcreteLogPersistence.

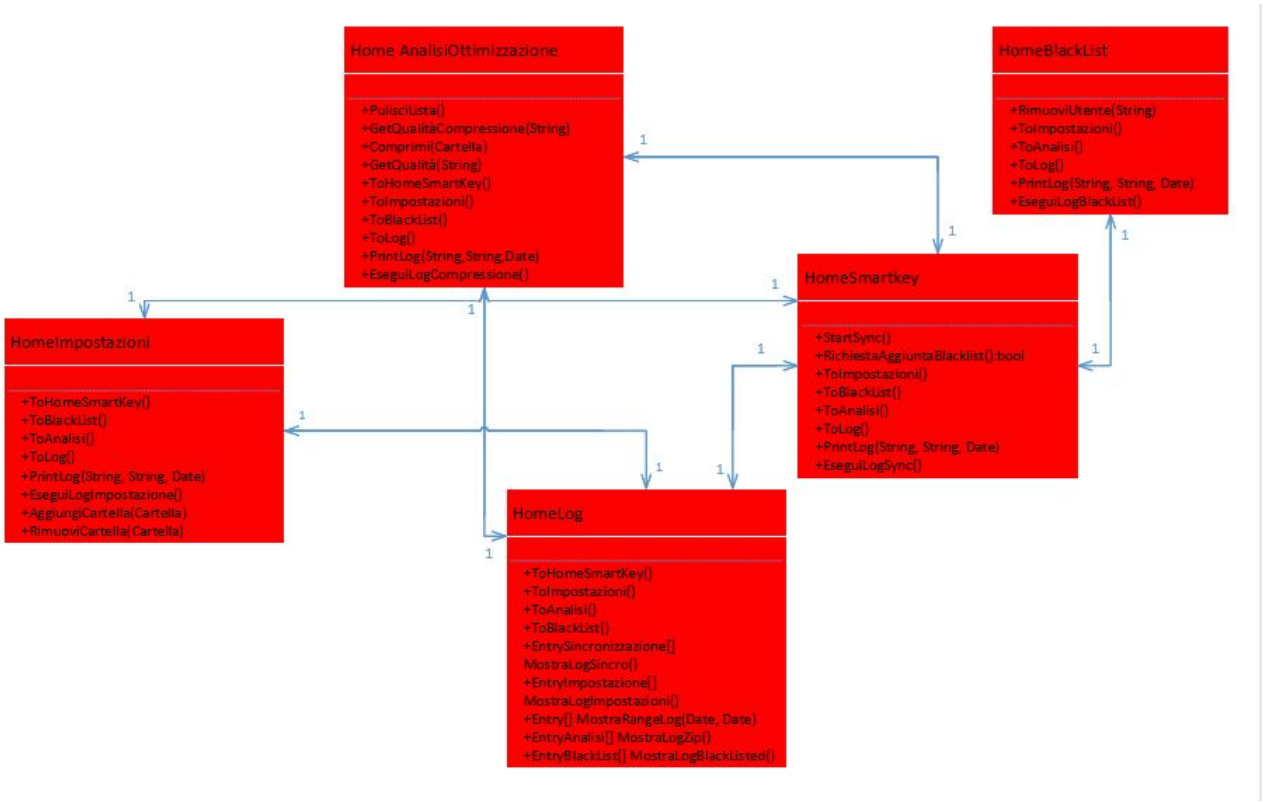
Diagramma di Dettaglio : InitControllers e DataPersistence



Si è scelto di astrarre molto la parte di persistenza per fare in modo che in futuro sia possibile passare da un tipo data source a un altro senza modificare l'architettura esistente, come richiesto per gli sviluppi futuri.
La classe InitControllers attraverso il metodo Init si occuperà di popolare le nostre strutture dati dalla persistenza, in particolare caricherà i file di persistenza della BlackList e ImpostazioniTrasferimento.

Il metodo salva sarà l'event handler dell'evento ToPersist scatenato dai controller che vogliono rendere persistenti le loro azioni.

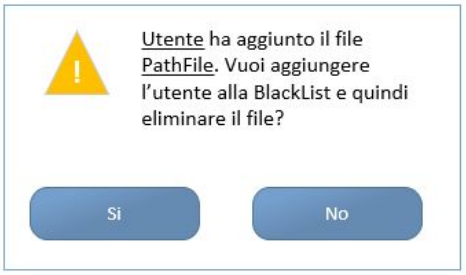
Diagramma di Dettaglio : View



Collegamenti View-Controller
Tutte le view sono gestite dalla classe MainController, fatta eccezione per la home che per la funzionalità di popup per la scelta della decisione dell'utente su chi inserire nella blacklist sarà delegata alla classe Controller per la sincronizzazione(GestoreSincronizzazioneController).

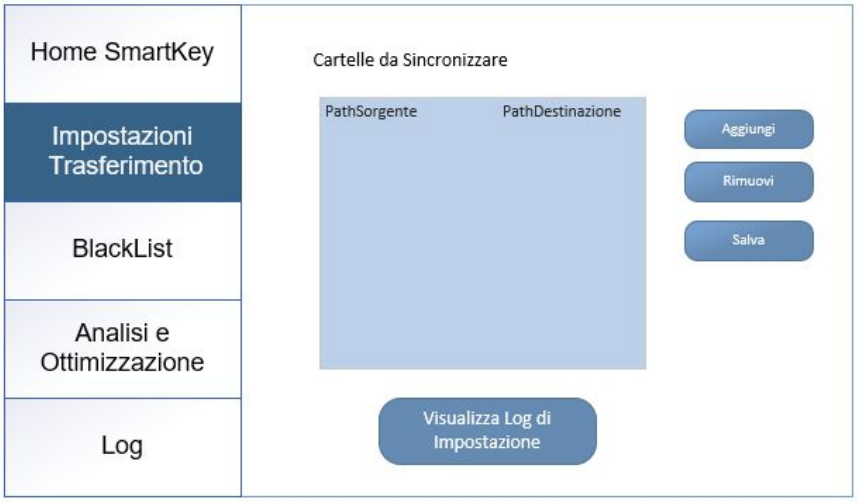
Nel diagramma di dettaglio delle View, per quanto riguarda la "View Log", abbiamo pensato di inserire la funzionalità di filtraggio delle entry in base al tipo di log registrato.
Di seguito riportiamo uno schema dei componenti delle varie interfacce

Diagramma di dettaglio: view Home SmartKey



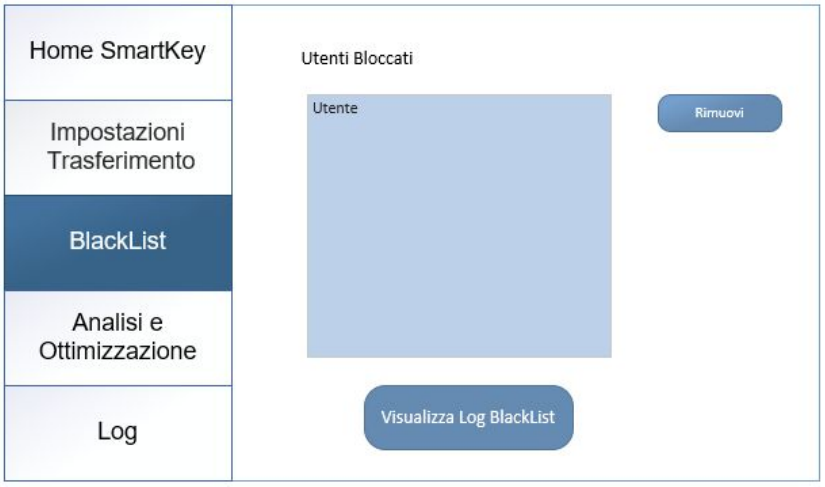
Questa sarà la schermata principale dell'applicazione, quella che si aprirà all'avvio.
Una volta premuto il tasto "Sincronizzazione", nel caso di aggiunta nella cartella pubblica dell'utente di un file di proprietà sconosciuta, verrà mostrato il popup per la scelta di aggiunta o meno dell'utente autore del file alla BlackList.
Sul lato sinistro sono presenti i bottoni per effettuare la navigazione tra le varie view.
Il pulsante sincronizza farà partire il servizio di sincronizzazione, mentre il pulsante "visualizza log di Sincronizzazione" andrà nella view del log (HomeLog) e imposterà i filtri in modo da ottenere solamente le entry del log che riguardano la sincronizzazione.

Diagramma di dettaglio: view Home Impostazioni Trasferimento



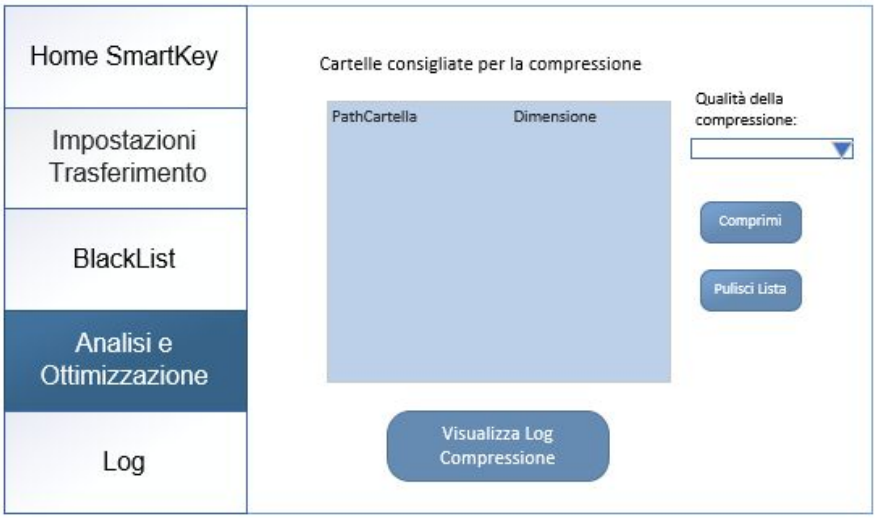
Questa view si occupa del mostrare le impostazioni di trasferimento, e offre i bottoni per inserirle, rimuoverle e salvare tutto in maniera persistente.
Quando premo il tasto “Aggiungi” si apre l’esplora risorse per scegliere la sorgente poi, una volta aggiunta, si apre in automatico un altro esplora risorse per la destinazione. Il verso della Sincronizzazione è, quindi, stabilito dalla coppia sorgente-destinazione
Anche questa view ha la possibilità di andare nella view di Log e impostare i filtri in maniera appropriata.

Diagramma di dettaglio: view HomeBlackList



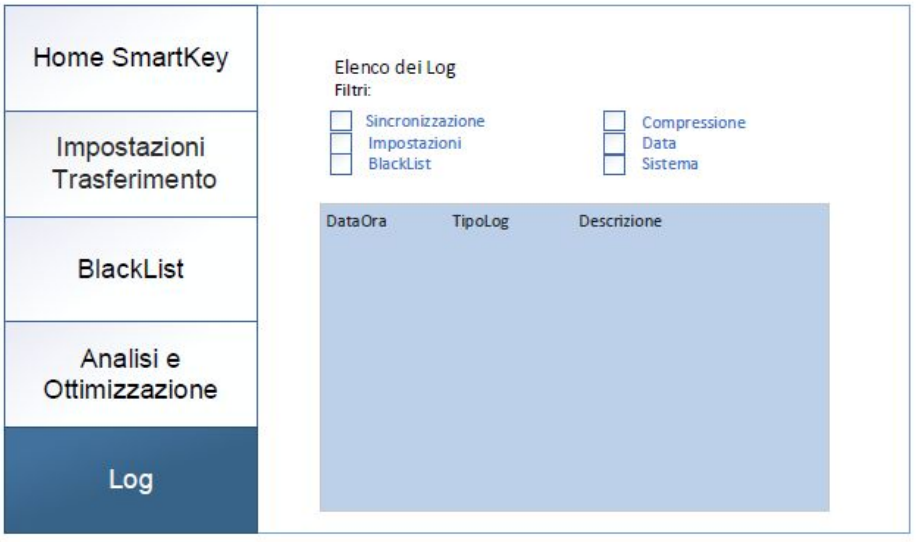
Questa view si occuperà di mostrare tutti gli utenti presenti nella blacklist di quell’utente e la possibilità di rimuoverli.
Anche questa view ha la possibilità di andare nella finestra di log e impostare in maniera corretta i filtri per il log.

Diagramma di dettaglio: view Analisi e Ottimizzazione



Questa è la view dedicata ad analisi e ottimizzazione dove è possibile scegliere tramite il menù a tendina la qualità della compressione, e di quali cartelle fare la compressione.
Anche questa view ha il bottone per andare nel file di log e impostare i filtri

Diagramma di dettaglio: view HomeLog



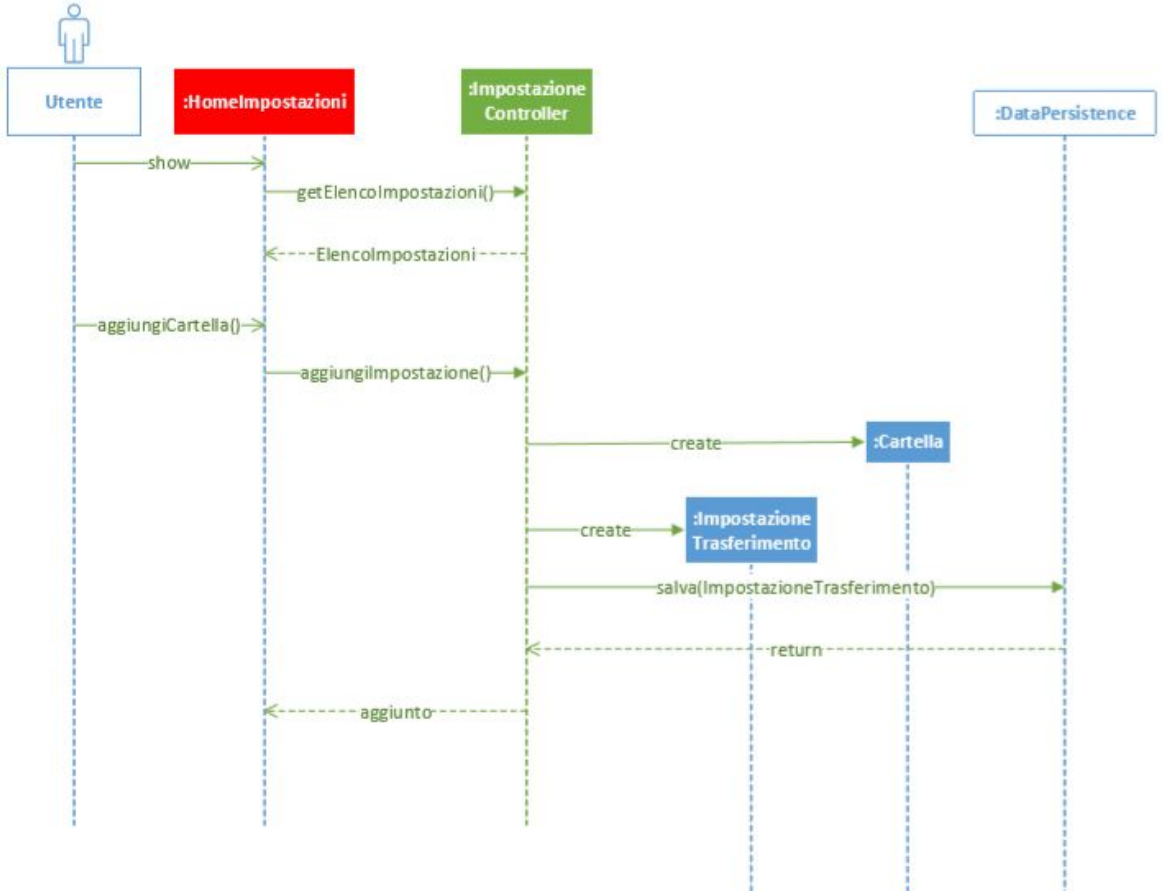
Questa è la schermata in cui si può visualizzare il file di log e impostare dei filtri in base ai quali verranno mostrate righe di log solo per quel tipo di azioni.
Alcune chiarificazioni:

- Una volta selezionata la checkbox “Data” compariranno 2 aree di input (aspetto grafico ancora di definire) in cui scegliere il range
- La checkBox “Sistema” mostra tutto ciò che riguarda anomalie del funzionamento e la protezione delle cartelle

Nella colonna “Descrizione” verranno riportate le informazioni come descritte nella sezione “formato log” riportata alla fine del documento.
Questa view viene ampiamente utilizzata dalle altre view.

Interazione

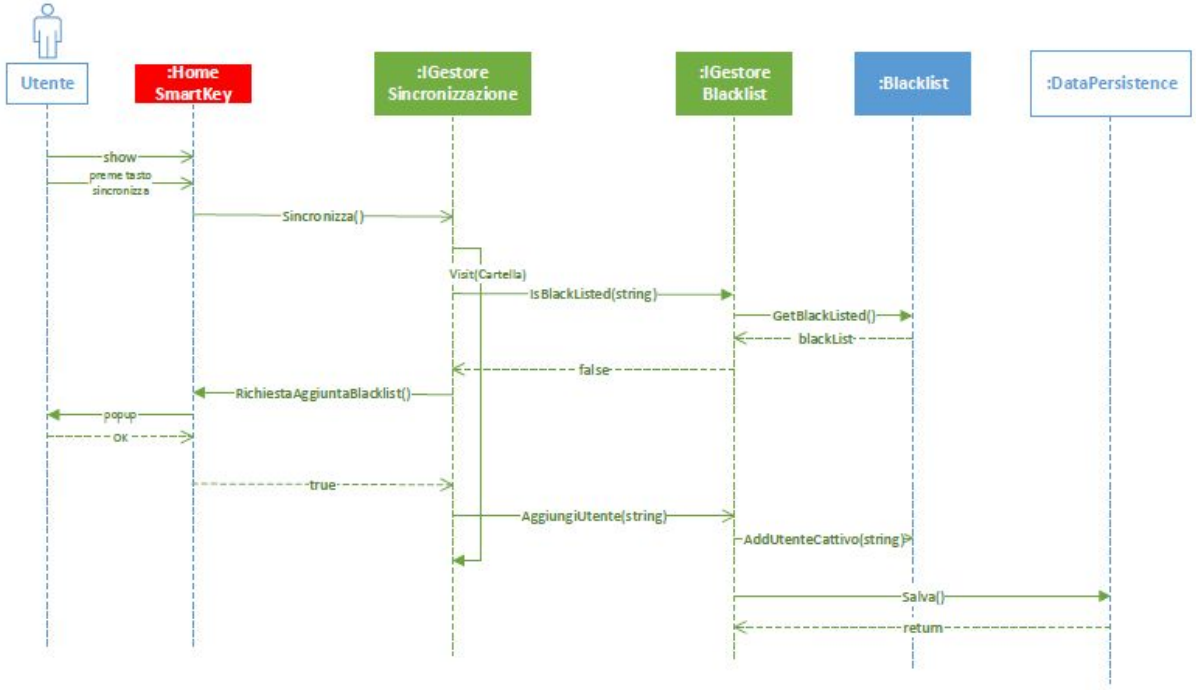
Diagramma di Sequenza : Aggiunta Impostazione



-

Il diagramma è simile a quello visto nell'Analisi del Problema, cambia la semantica di creazione dell'oggetto cartella e si è deciso di raggruppare le view per l'aggiunta dell'impostazione e quella per l'aggiunta della cartella in una unica "HomeImpostazioni".
Nel diagramma è riportata la creazione di un solo oggetto cartella, in realtà comporta sia la creazione dell'oggetto cartella ma anche per ogni elemento presente in quella cartella(file o cartella) viene creato un oggetto(FileWrapper o Cartella) e agganciato al proprio padre(pattern Composite).
Il metodo salva sarà chiamato grazie all'invocazione dell'evento ToPersist.

Diagramma di Sequenza : Aggiunta Utente in BlackList



Questo diagramma mostra come inizia la sincronizzazione e, supponendo che l'utente autore del file sconosciuto non sia in blacklist, viene richiesto all'utente tramite un popup se aggiungerlo o no e nel caso qui sopra abbiamo supposto una risposta positiva da parte dell'utente, quindi di aggiungerlo alla blacklist.
All'interno del metodo visit il comportamento cambierà a seconda della risposta dell'utente. Sono riportate le interazioni all'interno del metodo visit con il gestore della blacklist.

Comportamento

Non sono sviluppati ulteriori diagrammi delle attività: quello fatto in fase di analisi del problema risulta sufficiente.

Progettazione della Persistenza

La parte di persistenza è stata astratta in maniera tale da permettere il salvataggio su ogni piattaforma.
Come prima forma di persistenza abbiamo scelto un file di tipo XML, che sarà validato dalle seguenti grammatiche.

XMLSchema: ImpostazioneTrasferimento

Questo schema rappresenta come verranno salvate le impostazioni di trasferimento, sono stati introdotti vincoli sui dati anche a livello di validazione per evitare modifiche del file .xml inappropriate(i path devono essere lunghi al massimo 260 caratteri).
Anche il nome utente è sottoposto a un controllo come descritto nell'analisi del problema, non deve superare 256 caratteri.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Radice dei documenti XSD -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<!-- Definizione dell'intestazione della tabella -->
   <xs:simpleType name="path">
      <xs:restriction base="xs:string">
         <xs:maxLength value="260"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="user">
      <xs:restriction base="xs:string">
         <xs:maxLength value="256"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:complexType name="impostazioneTrasferimentoType">
      <xs:sequence>
         <xs:element name ="cartella-sorgente" type="path"/>
         <xs:element name="cartella-destinazione" type="path"/>
         <xs:element name="verso" type="xs:string"/>
      </xs:sequence>
      <xs:attribute name="utente" type="user" use="required"/>
   </xs:complexType>
   <xs:complexType name="impostazioniTrasferimentoType">
      <xs:sequence>
         <xs:element name="impostazione" type="impostazioneTrasferimentoType"  maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:element name ="impostazioni" type="impostazioniTrasferimentoType"/>
</xs:schema>
```

XMLSchema: Blacklist

Questo schema rappresenta come verranno salvate le varie blacklist (differenti per ogni utente) nel file XML.
Dato che in questo file sono presenti nome di utente, come in quello precedente, la lunghezza del nome dell'utente deve essere inferiore a 256 caratteri.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Radice dei documenti XSD -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:simpleType name="user">
      <xs:restriction base="xs:string">
         <xs:maxLength value="256"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:complexType name="blacklistType">
      <xs:sequence>
         <xs:element name="utente" type="user" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="proprietario" type="user" use="required"/>
   </xs:complexType>
   <xs:complexType name="blacklistsType">
      <xs:sequence>
         <xs:element name="blacklist" type="blacklistType" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:element name="blacklists" type="blacklistsType"/>
</xs:schema>
```

Formato File Log

Si riportano qui i formati delle entry con cui verranno salvate le varie operazioni nel file di log.

- Formato file per Log delle sincronizzazioni
  DataOra TipoEntry Operazione Sorgente Destinatario
- Formato file per Log di analisi e ottimizzazioni
  DataOra TipoEntry Operazione Sorgente
- Formato file per Log di impostazioni di trasferimento
  DataOra TipoEntry Operazione Sorgente Destinatario
- Formato file per Log della blacklist
  DataOra TipoEntry Operazione UtenteMalevolo
- Formato file per Log di sistema
  DataOra TipoEntry Operazione

Progettazione del Collaudo

```
class LogControllerTest
{
    private IGestoreBlacklist listController;
    private IGestoreImpostazione impostazioneController;
    private IObserver logController;
    0 riferimenti | 0 modifiche | 0 autori, 0 modifiche
    private void Init()
    {
        listController = new BlackListController();
        impostazioneController = new ImpostazioneTrasferimentoController();
        logController = new LogController();
        //Registro gli eventi
        listController.ToLog += logController.UpdateLog;
        impostazioneController.ToLog += logController.UpdateLog;

    }

    [TestCase]
    0 riferimenti | 0 modifiche | 0 autori, 0 modifiche
    public void TestLogImpostazione()
    {
        //Test che controlla che l'evento scatenato da add impostazione aggiunga una entry di tipo EntryImpostazione
        impostazioneController.AddImpostazione(new ModelGestione.ImpostazioneTrasferimento("mydir", "yourdir"));
        Assert.That(logController.Entries[logController.Entries.Count - 1 ] is EntryImpostazione);
    }

    [TestCase]
    0 riferimenti | 0 modifiche | 0 autori, 0 modifiche
    public void TestLogBlackList()
    {
        //Test che controlla che l'evento scatenato da aggiungiutente aggiunga una entry di tipo EntryBlacklist
        listController.AggiungiUtente("you");
        Assert.That(logController.Entries[logController.Entries.Count - 1] is EntryBlacklist);
    }
```

Progettazione del Deployment

Tutte le impostazioni del sistema sono configurabili attraverso il software SmartKey, si necessita quindi di una maschera per gestire l'aggiornamento a versioni future. Quindi è stata aggiunta un'interfaccia che permette all'utente di verificare e scaricare aggiornamenti del software SmartKey.

Implementazione

Problematiche

- Per garantire che la sincronizzazione fosse trasparente abbiamo dovuto effettuare tutte le operazioni dedicate alla sincronizzazione in un nuovo thread.
  Si sono però riscontrati dei problemi quando questo nuovo thread doveva aggiornare la view, tramite la manualistica di C# siamo riusciti a risolvere il problema.
- Purtroppo se non è presente un dominio comune (es. Active Directory), i nomi degli utenti della blacklist non hanno un nome significativo ma un GUID.
- L'oggetto Cartella si è rivelato fondamentale solamente per la rappresentazione della cartelle sorgenti(di impostazione trasferimento).
  Dato che la creazione di quest'oggetto è un pò dispendiosa si è pensato di limitarne l'utilizzo alla sola cartella sorgente.

Variazioni

Blacklist

- La classe Blacklist (modello) utilizza il pattern Singleton poiché al momento dell'esecuzione ci sarà solo una blacklist attiva e vogliamo un singolo punto di accesso.

Log

- La classe Entry (che rappresenta una riga del log) ha due costruttori: uno che prende la data attuale e uno che costruisce la entry in base alla data passata come parametro.
- Alla classe Entry abbiamo aggiunto un attributo "operazione" che rappresenta il tipo di operazione che si è effettuata.
- I metodi Getter e Setter presenti nelle varie Entry Specializzate non sono necessari pertanto non sono stati implementati.
- Alla classe EntryFactory è stato fatto overloading del metodo statico CreateEntry: un metodo per creare la entry con la data attuale e uno per poter ricreare la entry quando si effettua il parsing del file di log.
- Nella classe Log i getter e i setter sono stati implementati sotto forma di proprietà.

Impostazione Trasferimento

- L'attributo cartellaDestinazione nella classe ImpostazioneTrasferimento(modello) in fase di progettazione è stato modellato come un attributo di tipo Cartella, implementando il sistema ci si è resi conto che è più efficiente modellarlo come una stringa invece di ricreare anche tutto il sottoalbero per la cartella destinazione.
- Getter e setter nella classe di modello sono stati realizzati come Property.

- In ImpostazioneTrasferimentoController l'insieme delle impostazioni è stato rappresentato come un ISet e non come una IList, per garantire l'univocità delle impostazioni di trasferimento.

Sincronizzazione

- Nella classe GestioneSincronizzazioneController abbiamo aggiunto un attributo di tipo BackgroundWorker(Classe fornita dal linguaggio C#) che consente di effettuare la sincronizzazione in un thread separato
- Il metodo Sincronizza ha parametri di invocazione diversi perchè è diventato un event handler.
- È stato aggiunto un metodo privato InitSync per facilitare l'implementazione della funzione di sincronizzazione, scelta dovuta anche all'utilizzo della classe BackgroundWorker.
- GestioneSincronizzazioneController contiene un attributo di tipo IGestoreImpostazione per prendere le impostazioni di trasferimento.

Gestione delle view

- È stata aggiunta una classe per gestire la navigabilità tra le varie maschere del software SmartKey (Navigation Helper)

Collaudo

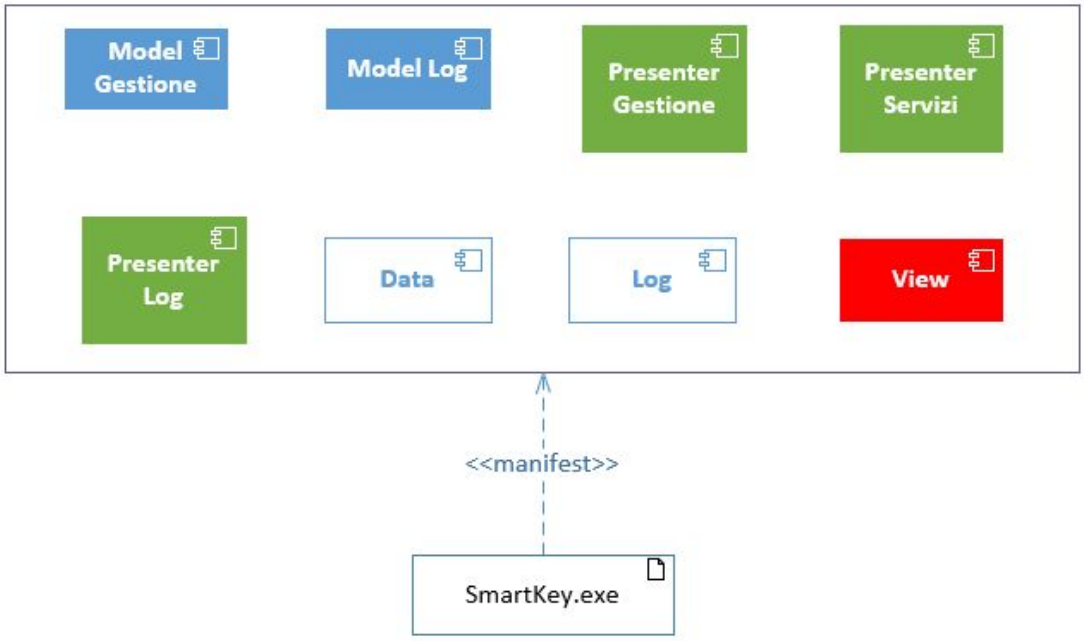
Il collaudo è stato effettuato sulle seguenti classi di test con esito positivo:

- LogControllerTest
- TestImpostazioniController
- TestLogSistema
- TestSincronizzazione
- BlacklistPersistenceXMLTest
- ImpostazioniTrasferimentoPersistenceXMLTest
- TestLogParsing
- TestLogPersistence
- TestFactory
- CartellaSubtreeTest
- CartellaTest
- FileWrapperTest
- BlacklistTest
- ImpostazioneTrasferimentoTest
- UtenteTest
- TestChiavetta

Deployment

Abbiamo inserito un diagramma di deployment, la nostra applicazione non è distribuita quindi il deployment avverrà su un unico nodo.

Artefatti



Deployment type-level